Prüfer:	Prof. Dr. rer. nat. Eggenberger
Betreuer:	Dipl.-Inf. Gawdi Azem

begonnen am:	15. April 2000
beendet am:	14. November 2000

CR-Klassifikation:	C.2.2, E.3, H.3.5

Diplomarbeit Nr. 1859

Entwurf und Realisierung einer Web-Oberfläche zum sicheren Fernzugriff auf Projektdaten

Stefan Gybas

Institut für Informatik
Universität Stuttgart
Breitwiesenstraße 20–22
70565 Stuttgart

Zusammenfassung

Ziel dieser Arbeit war die Konzeption und Implementierung einer Web-Oberfläche, die den Zugriff auf Netzlaufwerke mit einem normalen Webbrowser ermöglicht. Dabei sollen Dateien auf mehrerer verteilten Dateiservern heruntergeladen, hochgeladen oder gelöscht werden können. Falls weitere Operationen, wie zum Beispiel das Umbenennen von Dateien, benötigt werden, kann dies mit einem WebDAV-Client erreicht werden.

Die ersten beiden Kapitel beschreiben die Anforderungsanalyse, die Aufgabenstellung und den Grobentwurf. Grundlagen der objektorientierten Programmierung und der Unified Modeling Language (UML) werden dabei vorausgesetzt.

Das dritte Kapitel erläutert allgemein die eingesetzten Standards, Protokolle und Technologien. Im vierten Kapitel wird darauf aufbauend der Feinentwurf und die Implementierung des Projekts vorgestellt.

Das letzte Kapitel stellt schließlich das Handbuch für Benutzer und Administratoren des Systems dar. Sowohl zur Nutzung wie auch zur Konfiguration werden nur ein Webbrowser und Grundkenntnisse der Internet-Technologien benötigt.

Inhaltsverzeichnis

1 Analyse der Anforderungen **5**
- 1.1 Ist-Zustand (Umfeld) 5
- 1.2 Soll-Zustand 6
- 1.3 Aufgabenstellung und Vorgaben 7

2 Grobentwurf **9**
- 2.1 Anforderungen an das System 9
- 2.2 Aufteilung des Systems in Komponenten 11
- 2.3 Zu entwickelnde Komponenten 12

3 Verwendete Standards und Technologien **13**
- 3.1 Formulare in HTML 13
- 3.2 Cascading Style Sheets (CSS) 15
- 3.3 Hypertext Transfer Protocol (HTTP) 17
  - 3.3.1 Authentifizierung von Benutzern 18
- 3.4 WebDAV als Erweiterung zu HTTP 19
  - 3.4.1 Vorteile von WebDAV 20
  - 3.4.2 Neue Methoden in WebDAV 21
- 3.5 Verschlüsselung 23
  - 3.5.1 Kryptografische Grundlagen 23
  - 3.5.2 Kryptografische Authentifizierung 24
  - 3.5.3 Secure Sockets Layers (SSL) 25
  - 3.5.4 SSL-Verbindungen über HTTP-Proxys 27
- 3.6 Lightweight Directory Access Protocol (LDAP) 27
- 3.7 Common Internet File System (CIFS) 28
- 3.8 Extensible Markup Language (XML) 29
  - 3.8.1 Darstellung von HTML-Dokumenten als XML 29
  - 3.8.2 XSLT 30
  - 3.8.3 XML Schemata 32
- 3.9 Debian-Pakete 33
  - 3.9.1 Vorteile von Paketen 34

3.9.2 Erstellung von Debian-Paketen 35
3.10 Java-Standards 37
3.10.1 Java Beans 37
3.10.2 Java Servlets 39
3.10.3 JavaServer Pages (JSP) 40
3.10.4 XML mit Java 42
3.10.5 Web-Anwendungen 43

4 Implementierung und Test **47**
4.1 Überblick 47
4.2 Das WebDAV-Servlet 48
4.3 Zugriff auf CIFS-Laufwerke 51
4.4 Verwaltung der Konfigurationsdateien 54
4.5 Authentifizierung über LDAP 57
4.6 Aufteilung in Archive 59
4.6.1 Lizenzen der Pakete 60
4.7 Test des Systems 60
4.8 Zusammenfassung 63

5 Benutzung des Systems **65**
5.1 Benutzerhandbuch 65
5.1.1 Zugriff mit einem Webbrowser 65
5.1.2 Zugriff mit einem WebDAV-Client 68
5.2 Administration des Systems 71
5.2.1 Installation des Systems 71
5.2.2 Administration über die HTML-Oberfläche 74
5.2.3 Direkte Bearbeitung der Konfigurationsdatei 75

Literaturverzeichnis **77**

A Ursprüngliche Aufgabenstellung **83**

B GNU General Public License **85**
B.1 Vorwort 85
B.2 Allgemeine Öffentliche GNU-Lizenz 87
B.2.1 Keine Gewährleistung 92

C Erklärung **95**

Kapitel 1

Analyse der Anforderungen

Da große Teile dieser Analyse schon durch die Aufgabenstellung vorgegeben waren, fällt dieses Kapitel relativ knapp aus. Es erläutert nur die dortigen Angaben und zeigt die im Laufe des Projekts aufgetretenen und mit dem Betreuer abgestimmten Änderungen auf.

Teile dieses Kapitels gehen durch die Vorgaben des Betreuers schon in die Entwurfsphase über, zum Beispiel bei der Wahl der Programmiersprache und der eintzusetzenden Technologien.

1.1 Ist-Zustand (Umfeld)

In der Firma debis Systemhaus GEI mbH in Leinfelden-Echterdingen, arbeitet ein Teil der Mitarbeiter kurzzeitig oder über einen längeren Zeitraum von zu Hause aus oder bei einem Kunden. In beiden Fällen besteht zwar die Möglichkeit, sich über Modem oder ISDN in das interne debis-Netzwerk einzuwählen, für solche Fernzugriffe gelten jedoch strengere Richtlinien als für lokale Zugriffe.

Konkret bedeutet dies, dass sich solche Mitarbeiter in einem anderen Netzsegment als an ihrem Arbeitsplatz befinden. Dadurch ist kein direkter Zugriff auf Dateiserver am Arbeitsplatzes möglich, da solche Zugriffe durch eine Firewall unterbunden werden. Von diesem externen Netzsegment hat man nur Zugriff auf Rechner, die auch von anderen Standorten aus erreichbar sind und hierbei auch nur auf spezielle freigeschaltete Dienste, wie zum Beispiel einen Webserver.

Einerseits ist die Anzahl der IP-Adressen, die vom ganzen Konzern aus erreichbar sind, beschränkt, andererseits ist es nicht gewünscht, allen Mitarbeiter im gesamten debis Systemhaus Zugriff auf alle Dateiserver zu ermöglichen. Damit nun extern arbeitende Mitarbeiter doch Zugriff auf die Projektdaten auf den Dateiservern erhalten können, muss ein Umweg über eine Web-Schnittstelle gewählt werden.

1.2 Soll-Zustand

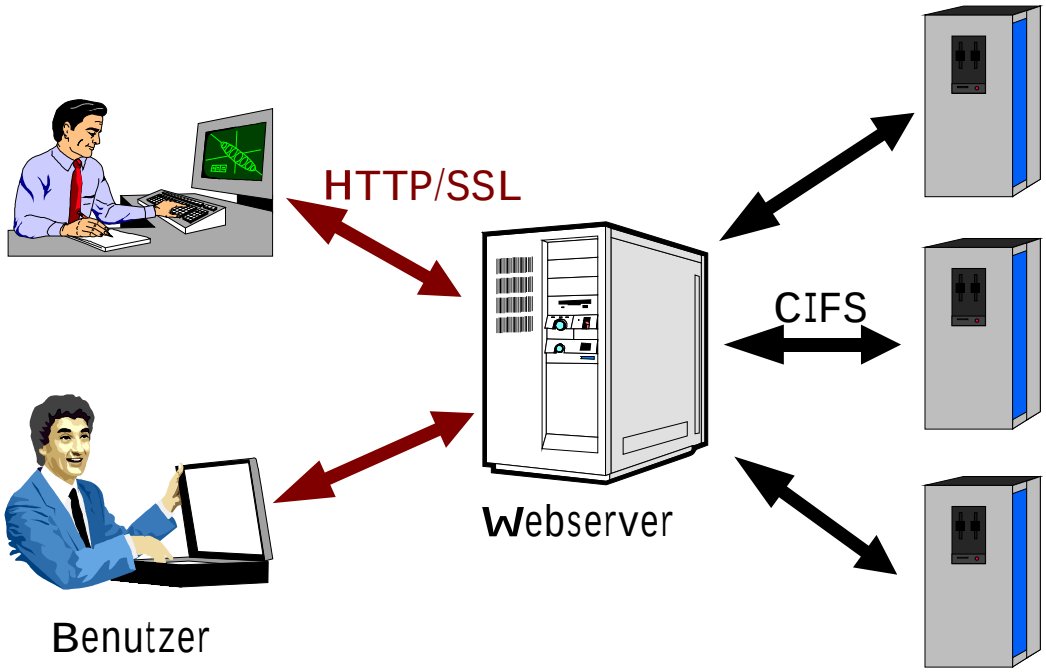
Nach Abschluss dieser Arbeit soll es extern arbeitenden Mitarbeitern ermöglicht werden, auf einen konzernweit zugänglichen Webserver zuzugreifen und so Zugang zu Projektdaten auf mehreren Dateiservern zu erhalten. Die TCP/IP-Verbindung vom Webbrowser der Mitarbeiter zum Webserver soll dabei nach dem SSL-Standard[1] verschlüsselt werden, damit vertrauliche Daten nicht von Unbefugten abgehört werden können.

Der Webserver bietet nach einer erfolgreichen Authentifizierung durch Benutzername und Passwort eine Liste aller für diesen Benutzer verfügbaren Netzlaufwerke an. Alle Zugriffe auf diese Netzlaufwerke werden dann vom Webserver über das CIFS-Protokoll[2] direkt an die entsprechenden Dateiserver weitergeleitet.

Wichtig ist hier, dass die Datenübertragung von den Benutzern zum Webserver verschlüsselt erfolgt, damit die Daten nicht auf ihrem Weg durch unkontrollierbare Netze abgehört werden können. Die Verbindungen zu den Dateiservern können auch verschlüsselt sein, dies ist aber nicht zwingend notwendig, da solch eine Kommunikation von den meisten Dateiservern nicht unterstützt wird. Vom Webserver aus besteht ein direkter Zugriff auf die Dateiserver, das heißt hier sind keine Firewalls mehr zwischengeschaltet.

Das folgende Diagramm veranschaulicht diesen Sachverhalt. Der Webserver ist dabei in der Mitte dargestellt und die einzelnen Dateiserver auf der rechten Seite. Eventuell auf der Strecke von den Benutzern zum Webserver vorhandene Proxy-Server wurden nicht eingezeichnet.

[1] Siehe Abschnitt 3.5
[2] Siehe Abschnitt 3.7



1.3 Aufgabenstellung und Vorgaben

Im Rahmen dieser Arbeit ist ein Programm zu erstellen, das dynamisch HTML-Seiten generiert. Dadurch reicht ein einfacher Webbrowser zum Zugriff auf Projektdaten aus. Alternativ kann der Zugriff auch über das WebDAV-Protokoll[3], wie es von Microsofts Internet Explorer ab Version 4.0 unterstützt wird, erfolgen. In jedem Fall ist darauf zu achten, dass keine spezielle, lokal zu installierende Software notwendig ist.

Im Gegensatz zur ursprünglichen Aufgabenstellung, die in Anhang A enthalten ist, sind in der HTML-Oberfläche nur Möglichkeiten zum Download, Upload und Löschen von Dateien notwendig. Alle anderen Aufgaben, wie zum Beispiel das Umbenennen von Dateien oder Anlegen von Verzeichnissen, können über das WebDAV-Protokoll erledigt werden.

Zusätzlich zur HTML-Oberfläche für Administratoren muss auch noch die Möglichkeit bestehen, Änderungen an den Einstellungen direkt in einer Konfigurationsdatei vorzunehmen. Daher ist es nicht möglich, die Konfigurati-

[3]Eine ausführliche Beschreibung des WebDAV-Protokolls ist in Abschnitt 3.4

onsdaten in Datenbanken zu speichern. Es muss ein Datenformat gewählt werden, das mit jedem beliebigen Texteditor bearbeitet werden kann. Als Standard bietet sich hier XML[4] an, zumal es zum Erstellen und Einlesen solcher Dateien schon fertige Programmbibliotheken gibt.

Die Möglichkeit zur Erstellung von Sicherheitskopien von Projektdaten, die über dieses Programm verändert oder gelöscht werden, wird dahingehend abgeändert, dass auf einem separaten lokalen Datenspeicher nur eine Kopie der alten Version mit angehängtem Zeitstempel gespeichert werden muss. Die Möglichkeit zur Darstellung der Änderungen zwischen zwei Versionen wird nicht verlangt.

Das gesammte Programm soll einfach auf weiteren Rechnern zu installieren sein. Es kann hierbei davon ausgegangen werden, dass auf den betreffenden Rechnern ein Debian GNU/Linux-System installiert ist, so dass es sich anbietet, das Programm als Debian-Paket[5] zur Verfügung zu stellen.

[4] XML wird in Kapitel 3.8 erläutert
[5] Siehe Abschnitt 3.9

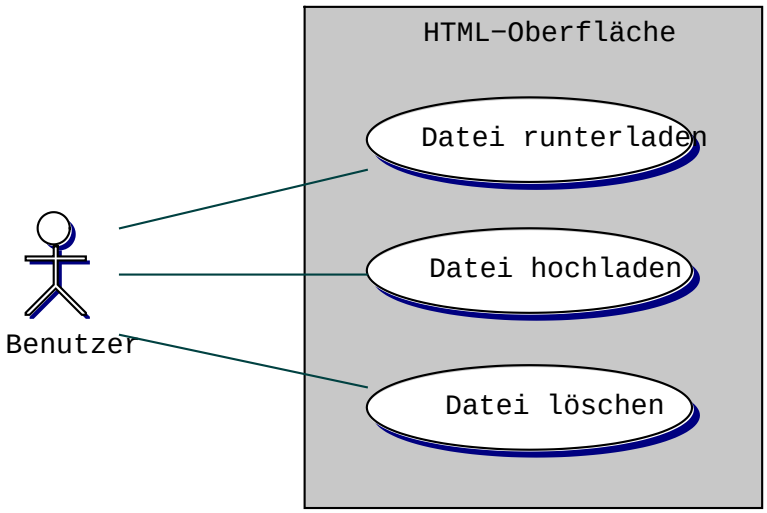
Kapitel 2

Grobentwurf

2.1 Anforderungen an das System

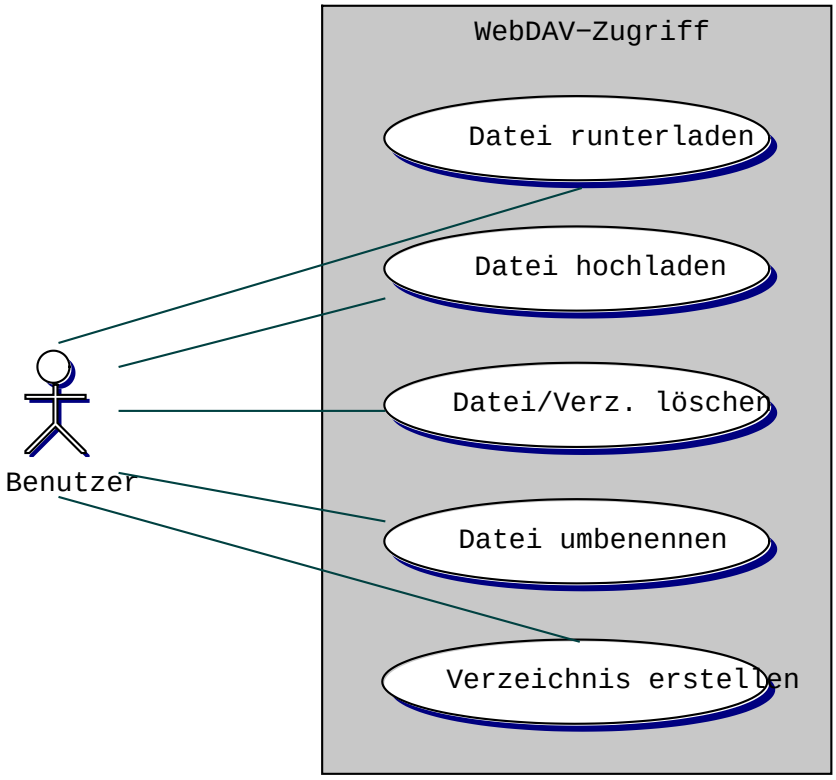
Benutzer des Systems haben die Wahl zwischen zwei Zugriffsmöglichkeiten. Die technischen Grundlagen der verwendeten Protokolle werden im nächsten Kapitel näher erläutert.

1. Die erste Art besteht aus dem Zugriff mit einem normalen Webbrowser auf eine HTML-Oberfläche mit den nachfolgend dargestellten Anwendungsfällen als UML-Diagramm:



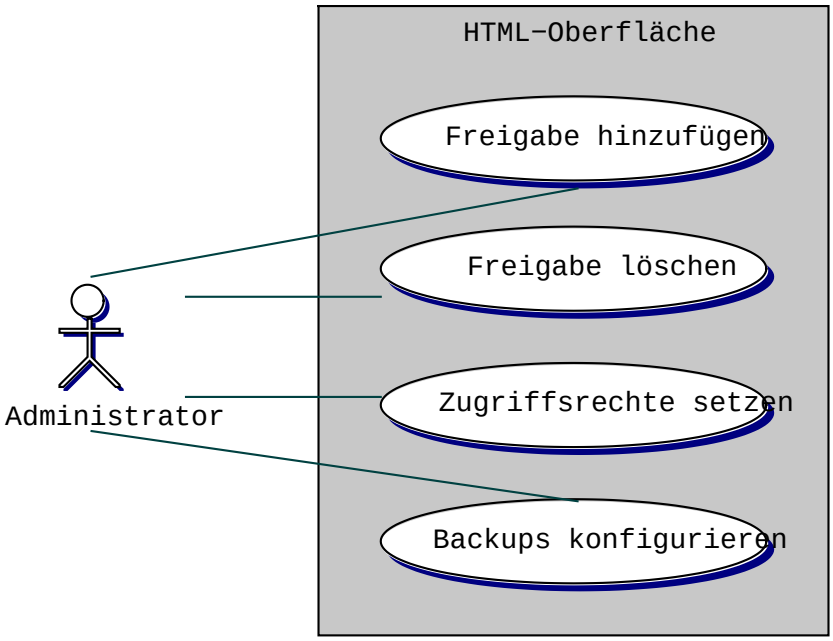
Das Löschen von Dateien soll dabei durch Markieren aller zu löschenden Dateien und anschließendem Betätigen eines entsprechend markierten Knopfs eines HTML-Formulars geschehen. Zum Hochladen (engl. upload) von Dateien soll ebenfalls ein Knopf verwendet werden, nach dessen Betätigung ein Dialog zur Auswahl der hochzuladenden Datei(en) erscheint.

2. Die zweite Zugriffsmöglichkeit geschieht mit Hilfe eines WebDAV-Clients, der gegenüber der erste Methode weitere Operationen zulässt:



WebDAV-Clients bieten eine eigene grafische Oberfläche, die meist der des Windows-Explorers nachempfunden ist. Außerdem ist es mit aktuellen Versionen von Microsoft Windows auch möglich, auf WebDAV-Verzeichnisse direkt mit dem Explorer zuzugreifen.

Als letzte nach außen sichtbare Schnittstelle ist für Administratoren noch die Möglichkeit zur Konfiguration des Systems über Webbrowser zu realisieren. Diese Methode kann alternativ zur direkten Bearbeitung der XML-Konfigurationsdateien gewählt werden:

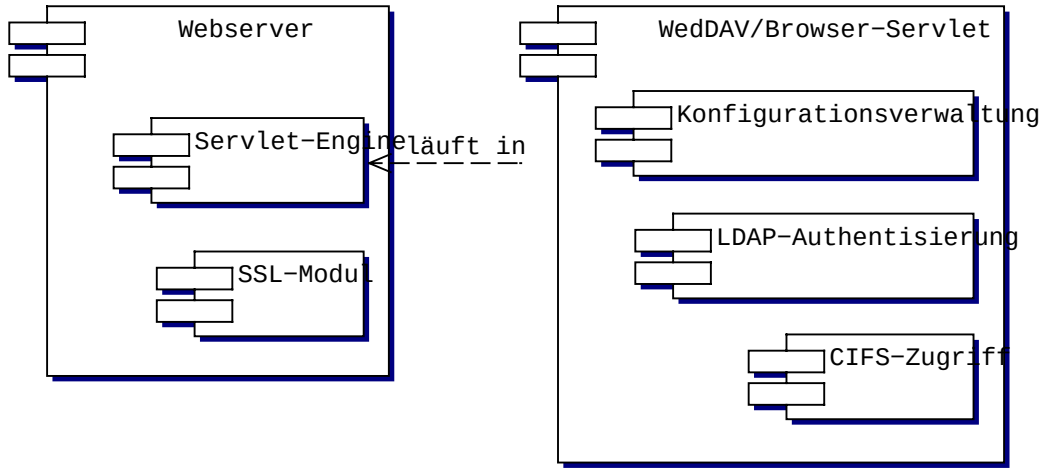


Änderungen, die mit dieser Schnittstelle vorgenommen werden, müssen sich sofort auswirken. Dies bedeutet, dass zum Beispiel ein neues Netzlaufwerk für alle folgenden Zugriffe verfügbar ist, ohne den Webserver neu zu starten.

2.2 Aufteilung des Systems in Komponenten

Das Komplettsystem besteht im Wesentlichen aus einem Webserver, der die dynamisch generierten Seiten an die Clients weiterleitet. Zur Verschlüsselung der Übertragung ist es notwendig, dass der Webserver den SSL-Standard unterstützt. Außerdem muss er die Möglichkeit bieten, Java Servlets, die mit der Java-Version JDK 1.2 erstellt wurden, einzubinden.

Die gesamte zu erstellende Anwendung besteht aus nur einem Servlet, das für jede Anfrage über den Webserver aufgerufen wird. Das Servlet selber besteht aus mehreren Komponenten, wie im folgenden Diagramm dargestellt:



Die einzelnen Komponenten werden im Kapitel 4.1 („Implementierung und Test") im Detail erläutert.

2.3 Zu entwickelnde Komponenten

Für den Webserver mit SSL-Modul und Servlet-Engine kann auf bestehende Produkte zurückgegriffen werden. Gerade in diesem Bereich gibt es frei erhältliche Open Source-Projekte, deren Leistungsfähigkeit sich gut mit kommerziellen Produkten messen lassen kann. Die tatsächlich im Rahmen dieser Arbeit ausgewählten Komponenten sind aus Kapitel 4.1 ersichtlich.

Es bleibt also die Implementierung des Servlets samt seiner im vorigen Kapitel dargestellten Komponenten. Teilweise kann auch hier auf bereits erhältliche Open Source-Software zurückgegriffen werden. Zum Beispiel gibt es für den Zugriff auf CIFS-Netzlaufwerke schon fertige Klassen, deren Schnittstelle jedoch an die Anforderungen angepasst werden musste. Auch zum Einlesen und Schreiben der XML-Konfigurationsdateien existieren, wie bereits im vorigen Kapitel erwähnt, direkt verwendbare Klassen.

Die hierbei im Einzelnen eingesetzten Standards und Technologien werden im nächsten Kapitel behandelt. Detailliertere Informationen und die kompletten Spezifikationen befinden sich in den im Literaturverzeichnis angegebenen Quellen.

Kapitel 3

Verwendete Standards und Technologien

3.1 Formulare in HTML

HTML, die Hypertext Markup Language, ist eine so genannte Auszeichnungssprache (engl. markup language), die im Internet zur Gestaltung von Webseiten eingesetzt wird. Mit ihrer Hilfe wird die logische Struktur eines Dokuments beschrieben, also Teile wie Überschriften, Tabellen und Verweise auf andere Dokumente. Ursprünglich war es nicht vorgesehen, das genaue Layout eines Dokuments mit HTML festzulegen, spezielle Erweiterungen von Netscape und Microsoft haben später aber auch dies ermöglicht.

Ein HTML-Dokument besteht aus ASCII-Text und so genannten Steuerbefehlen (engl. tags), die durch spitze Klammern markiert werden. Das gesamte Dokument besteht also nur aus reinem Text und enthält keine Binärdaten, was die Bearbeitung mit jedem beliebigen Texteditor ermöglicht und die dynamische Erzeugung in Programmen vereinfacht.

Es gibt zwei Arten von Tags, die optional auch Attribute besitzen dürfen:

- Container-Tags bestehen aus einem Anfangs- und Ende-Tag, zwischen denen beliebiger Text oder auch weitere Tags stehen dürfen. Durch diese Schachtelung ergibt sich eine baumartige Struktur eines jeden HTML-Dokuments.
- Die zweite Art sind einzelne Tags ohne zugehöriges Ende-Tag, die vor allem zum Einfügen von Zeilenumbrüchen und Bildern verwendet wer-

den, die Quelle des Bildes wird in letzterem Fall als Attribut angegeben.

Das folgende Beispiel stellt ein minimales HTML-Dokument mit Typangabe, Einbindung einer Stilvorlage (siehe Abschnitt 3.2), einer Überschrift, einem Textabschnitt, einem Kommentar und einem Bild dar:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
  <head>
    <link rel=stylesheet type="text/css" href="style1.css">
    <title>Einfaches HTML-Beispiel</title>
  </head>

  <body>
    <h1>Überschrift des Beispiels</h1>

    <p>
      Hier ein Textabsatz. Er kann natürlich mehrere Sätze
      enthalten.
    </p>

    <!-- Dies ist ein Kommentar, als nächstes wird ein
         JPEG-Bild in das aktuelle Dokument eingebunden. -->

    <img alt="" height=80 width=80 src="bild1.jpg">
  </body>
</HTML>
```

Eine Liste aller erlaubten und notwendigen Tags ist in der Spezifikation der aktuellen HTML-Version 4.01 in [HTML 4.01] enthalten. Eine gute Einführung in die HTML-Programmierung ist [Münz], das auch kostenlos über das Internet gelesen werden kann.

Entworfen wurde HTML schon Ende der achtiger Jahre am CERN in Genf zur Präsentation von wissenschaftlichen Dokumenten. Schnell kam aber der Wunsch der Benutzer auf, nicht nur statische Dokumente lesen zu können, sondern auch interaktiv deren Inhalt beeinflussen zu können. Ein wichtiges und auch das heute am häufigsten eingesetzte Anwendungsgebiet ist die Abfrage von Datenbanken mit Hilfe eines Webbrowsers: Der Benutzer füllt ein Formular aus, schickt die eingegebenen Werte an einen Webserver und erhält eine dynamisch erzeugte HTML-Seite als Antwort zurück.

Zur Eingabe von Informationen in Formulare stehen verschiedene Elemente zur Verfügung:

- Textinformation, wie zum Beispiel ein Name oder eine Anschrift, können in Textfeldern oder mehrzeiligen Textboxen eingegeben werden.
- Zur Auswahl von mehreren Möglichkeiten, zum Beispiel der Versandart bei einer Online-Bestellung, gibt es Auswahllisten oder so genannte Radioknöpfe (engl. radio buttons), von denen nur einer ausgewählt werden kann.

Neuere HTML-Versionen, wie die aktuelle Version 4.0 (siehe [HTML 4.01]), unterstützen neben den traditionellen Eingabemethoden auch das Hochladen von Dateien. Erst dadurch wird es möglich, einen vollständigen Dateimanager in einem Webbrowser zu implementieren.

Das folgende Beispiel stellt einen Ausschnitt aus einer HTML-Datei mit einem Formular dar:

```
<form method="post" action="login.jsp">
  <p>Name: <input type="text" name="name" size=20></p>
  <p>Passwort: <input type="password" name="pass" size=20></p>
  <p>Server:
    <select name="server" size="1">
      <option> tick
      <option> trick
      <option> track
      <option> rupert
    </select>
  </p>
  <p><input type="submit" value="Einloggen"></p>
</form>
```

3.2 Cascading Style Sheets (CSS)

Um dem ursprünglichen Gedanken von HTML, der Trennung von Struktur und Layout, wieder gerecht zu werden, wurde eine eigene Sprache zur Layoutgestaltung von Webseiten entwickelt. Mit Hilfe dieser Sprache lassen sich Formatvorlagen (engl. style sheets) definieren, die beliebig ineinander geschachtet werden können. Dadurch kann das Layout einer Webseite von

Grafikdesignern erstellt werden, während der eigentliche textuelle Inhalt von Redakteuren oder dynamisch von einem Programm[1] erzeugt wird.

Durch die Möglichkeit der Schachtelung von Formatvorlagen lässt sich ein firmenweites Basislayout erstellen, das bei Bedarf auch für einzelne Seiten angepasst werden kann. Betrachter der Webseiten erhalten so ein konsistentes Layout, zum Beispiel immer die gleiche Hintergrundfarbe und Schriftart.

Formatvorlagen lassen sich aber auch zur individuellen Gestaltung von automatisch generierten Daten verwenden. So kann zum Beispiel immer das Firmenlogo im Hintergrund platziert werden.

Spezifiziert sind die Formatvorlagen für Webseiten durch die so genannten Cascading Style Sheets (CSS) Version 1.0 in [CSS 1.0]. Mit ihrer Hilfe lässt sich das Layout jedes HTML-Elements genauestens bestimmen. Es lassen sich die Schriftart, die Schriftgröße, die Farbe der Schrift, die Hintergrundfarbe, Abstände sowie Schriftattribute, wie zum Beispiel kursiv, festlegen. CSS-Spezifikationsdateien sind, wie auch HTML-Dateien, reine Textdateien, die sich mit jedem beliebigen Texteditor bearbeiten lassen. Hauptsächlich werden sie aber mit grafischen Editoren von professionellen Designern erstellt.

Das folgende Beispiel demonstriert die Layoutgestaltung einiger HTML-Tags, wie sie für die Darstellung von HTML 2.0 empfohlen wurden:

```
H1, H2, H3, H4 { margin-top: 1em; margin-bottom: 1em }
H5, H6 { margin-top: 1em }
H1 { text-align: center }
H1, H2, H4, H6 { font-weight: bold }
H3, H5 { font-style: italic }

H1 { font-size: xx-large }
H2 { font-size: x-large }
H3 { font-size: large }

B, STRONG { font-weight: bolder }  /* relative to the parent */
I, CITE, EM, VAR, ADDRESS, BLOCKQUOTE { font-style: italic }
PRE, TT, CODE, KBD, SAMP { font-family: monospace }

PRE { white-space: pre }
```

[1] In Java gibt es eine einfache Möglichkeit, solche Programme zu erstellen. Sie werden Servlets genannt und in Abschnitt 3.10.2 beschrieben.

3.3 Hypertext Transfer Protocol (HTTP)

Zur Übertragung der Daten, die in einem Webbrowser dargestellt werden sollen, wird im Internet das Hypertext Transfer Protocol (HTTP) eingesetzt. Hierbei wird vom Client eine TCP/IP-Verbindung zum Webserver aufgebaut, innerhalb derer zuerst die Anfrage des Clients und danach die Antwort des Servers übertragen werden.

Die Anfrage besteht aus beliebig vielen Kopfzeilen (engl. header lines), danach kommt in einer Zeile die Übertragungsmethode gefolgt vom lokalen Ort des gewünschten Dokuments und der Protokollversion, jeweils durch Leerzeichen getrennt. Abgeschlossen wird die Anfrage durch eine Leerzeile.

Die beiden am häufigsten eingesetzten Übertragungsmethoden sind `GET` zur Anforderung von einzelnen Dateien und `POST` zur Übermittlung der Inhalte eines Formulars. Wenn man sich noch einmal das Beispiel aus Abschnitt 3.1 ansieht, stellt man fest, dass in dem `<form>`-Tag genau diese Übertragungsmethode gewählt wurde. Die genaue Kodierung der Formulardaten kann dem HTTP-Standard unter [RFC 2616] entnommen werden.

Eine konkrete Anfrage, wie sie vom Netscape Communicator 4.75 zur Anforderung der Datei index.html geschickt wird, sieht folgendermaßen aus:

```
User-Agent: Mozilla/4.75 [en]
Accept: image/gif, image/jpeg, image/png, */*
Accept-Encoding: gzip
Host: localhost
Accept-Charset: iso-8859-1,*,utf-8
Referer: http://localhost/doc/index.html
GET /index.html HTTP/1.0
```

Nachdem der Webserver die Leerzeile zum Beenden der Anfrage gelesen hat, beginnt er mit der Verarbeitung der Daten und schickt anschließend seine Antwort innerhalb der selben HTTP-Verbindung an den Client zurück. Dabei teilt er ihm zuerst einen Statuswert, dann den Typ der Daten und in den meisten Fällen auch die Länge mit. Der Dateityp wird im MIME-Standard[2] kodiert und lautet für gewöhnliche HTML-Dateien `text/html`. Auf die vorherige Anfrage wird also eine Antwort der folgende Art gesendet, die zuerst

[2]Der MIME-Standard ist in [RFC 2045] spezifiziert.

aus einer Header, einer Leerzeile und schließlich den eigentlichen Nutzdaten besteht:

```
HTTP/1.1 200 OK
Date: Wed, 01 Nov 2000 17:13:41 GMT
Server: Apache/1.3.12 (Unix)
Last-Modified: Wed, 24 Mar 1999 02:35:47 GMT
Accept-Ranges: bytes
Content-Length: 3829
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
[Inhalt der HTML-Datei]
```

3.3.1 Authentifizierung von Benutzern

Das Hypertext Transfer Protocol bietet neben der Übertragung von Daten auch die Möglichkeit zur Authentifizierung von Benutzern. In diesem Fall wird vom Client eine zusätzliche Kopfzeile in die Anfrage eingefügt, nachdem der Server zuvor eine Authentifizierung gefordert hat. Hierbei gibt es zwei Arten:

1. Die einfachere Art überträgt den Benutzernamen und das zugehörige Passwort fast im Klartext über das Netzwerk. Diese beiden Daten werden lediglich durch einen Doppelpunkt getrennt zusammengefügt und nach dem Base64-Standard, wie in [RFC 2617] beschrieben, kodiert.

   Bei dieser, Basic-Authentifizierung genannten Methode, hat der Webserver die Möglichkeit, das Passwort wieder im Klartext zu gewinnen. Es kann dann zum Beispiel von einem Skript, das dynamische Webseiten erstellt, zum Login bei einem Datenbankzugriff verwendet werden.

   Diese Methode hat aber auch den Nachteil, dass jeder, der die HTTP-Verbindung mithört, unberechtigten Zugang zu dem Benutzernamen und dem Passwort erlangen kann. Solch ein Angriff kann an jedem Netzwerkknoten oder jeder Leitung auf dem Weg vom Client zum Server erfolgen.

2. Die Digest-Authentifizierung hingegen verwendet einen sicheren Hash-Algorithmus und sendet über das Netzwerken nur eine Prüfsumme, aus der das unverschlüsselte Passwort nicht mehr ermittelt werden kann.

   Damit der Webserver aber dennoch feststellen kann, ob der Benutzer das korrekte Passwort eingegeben hat, muss er lokal das Passwort im Klartext gespeichert haben. Er wendet auf dieses Passwort den selben Hash-Algorithmus an und vergleicht das Ergebnis mit dem vom Client übermittelten Wert.

   In diesem Fall ist es nicht möglich, auf externe Authentifizierungsverfahren, zum Beispiel den Verzeichnisdienst LDAP[3], zurückzugreifen, da man hierfür in der Regel das unverschlüsselte Passwort benötigt.

Da bei der zu implementierenden Web-Oberfläche auf Daten von externen Dateiservern zugegriffen werden muss, kann hier nur die Basic-Authentifizierung verwendet werden. Die Authentifizierung bei diesen Dateiservern geschieht zwar über ein verschlüsselt übertragenes Passwort, aber der Verschlüsselungsalgorithmus ist mit dem der Digest-Authentifizierung nicht identisch. Der Webserver ist daher darauf angewiesen, das Passwort des Benutzers im Klartext zu kennen, da er es selber nicht entschlüsseln und mit einem anderen Verfahren neu verschlüsseln kann.

Damit nun auf der Verbindungsstrecke zwischen Webbrowser und Webserver das Passwort nicht mitgehört werden kann, wird dringend empfohlen, hier nur eine verschlüsselte Verbindung nach dem SSL-Standard einzusetzen. Details dieser Verschlüsselungsart werden im Abschnitt 3.5 behandelt.

3.4 WebDAV als Erweiterung zu HTTP

Wie im vorherigen Abschnitt schon erläutert, wird bei einer Anfrage nach dem HTTP-Standard eine Übertragungsmethode angegeben. In den meisten Fällen lautet diese `GET` oder `POST`, es besteht hier aber die Möglichkeit, HTTP beliebig zu erweitern und neue Methoden einzuführen. Genau dieser Weg wird bei WebDAV, wie in der Spezifikation unter [RFC 2518] beschrieben, gegangen.

WebDAV ermöglicht nicht nur den Download von Dateien, sondern bietet darüber hinaus noch einige Erweiterungen:

[3] Siehe Abschnitt 3.6

- Upload von Dateien[4]
- Standardisiertes Auslesen und Bearbeiten von Eigenschaften (engl. properties) einer Datei in Form von XML-Bäumen. Dazu gehören neben den schon in Dateisystemenen verwendeten Attributen, wie zum Beispiel das Datum der letzten Änderung, auch der Autor und die Versionsnummer.
- Managementfähigkeiten für Dateien und Verzeichnisse[5], wie das Löschen, Umbenennen, Kopieren, Verschieben und Anlegen
- Einrichtung von Zugriffssperren (engl. locks), die Änderungen von anderen Benutzern verhindern, solange man selbst eine Datei bearbeitet
- Verwaltung mehrerer Versionen eines Dokuments mit Anzeige der durchgeführten Änderungen zwischen zwei Versionen

Gedacht ist dieses System zur Verwaltung von Webbereichen, in dem bestehende HTML-Seiten zuerst heruntergeladen werden, dann lokal mit einem Text- oder grafischen HTML-Editor bearbeitet und schließlich wieder hochgeladen werden. WebDAV stellt also ein mächtiges System zur Verwaltung von Dokumenten aller Art in einer hierarchischen Verzeichnisstruktur dar.

3.4.1 Vorteile von WebDAV

Der gesamte WebDAV-Standard nutzt zur Übermittlung sämtlicher Nutzdaten das XML-Format[6]. Dadurch ist es möglich, WebDAV mit allen gängigen Programmiersprachen zu verwenden, da sowohl HTTP- wie auch XML-Bibliotheken frei im Internet zur Verfügung stehen. Ein weiterer Vorteil ist die einfache Erweiterbarkeit: Es können jederzeit neue XML-Tags standardisiert werden, die dann von alten Programmen einfach ignoriert werden.

Gegenüber anderen Protokollen zur Übertragung von Dateien, wie zum Beispiel FTP nach [RFC 959], hat WebDAV mehrere entscheidende Vorteile:

1. Sichere Authentisierung von Benutzern durch die Verwendung des digest-Verfahrens. Dieses Verfahren ist im WebDAV-Standard für unverschlüsselte Übertragungen vorgeschrieben, damit das Passwort nicht

[4] HTTP 1.1 definiert zwar bereits eine `PUT`-Methode zum Upload von Dateien, in der Praxis wird diese jedoch nicht verwendet.

[5] Verzeichnisse werden im WebDAV-Standard als Sammlung (engl. collection) bezeichnet.

[6] Siehe Abschnitt 3.8

mitgehört werden kann.

2. Verschlüsselung sämtlicher übertragener Daten, also auch der Steuerinformationen, durch die Verwendung von SSL (siehe Abschnitt 3.5)
3. Unterstützung von Proxys, die die Anfrage für den Client an den Webserver weiterleiten. Dadurch ist der Aufbau sicherer Firewalls am Übergang von einem Firmennetz zum Internet möglich, ohne direkte TCP/IP-Verbindungden erlauben zu müssen.
4. Unterstützung von Caches, die meist in Kombination mit einem Proxy realisiert werden und das Zwischenspeichern von Dokumenten erlauben, damit sie bei einem erneuten Zugriff eines anderen Benutzern nicht noch einmal über ein eventuell langsames Weitverkehrsnetz übertragen werden müssen.
5. Effizienteres Protokoll, das mit einer einzigen TCP/IP-Verbindung auch mehrere Dateien übertragen kann. Dies ist auch schon mit HTTP 1.1 möglich und wird dort als Keep Alive-Verbindung bezeichnet.

3.4.2 Neue Methoden in WebDAV

Für alle oben genannten Anwendungsgebiete wurden neben neuen Feldern im Kopf einer Anfrage auch neue Methoden eingeführt. Weiterhin ist es auch möglich, Nutzdaten im XML-Format nach der eigentlichen HTTP-Anfrage zu übermitteln, zum Beispiel zur Spezifizierung der gewünschten Eigenschaften bei der `PROPFIND`-Methode.

Die neu eingeführten Methoden mit einer kurzen Beschreibung zeigt die folgende Tabelle:

`DELETE`	Löschen einer Datei oder eines *leeren* Verzeichnisses
`COPY`, `MOVE`	Kopieren bzw. Löschen einer Ressource[7]
`LOCK`, `UNLOCK`	Anlegen und Entfernen einer Sperre auf eine Ressource
`PROPFIND`	Anzeigen der Dateien und ihrer Eigenschaften innerhalb eines Verzeichnisses
`PROPPATCH`	Ändern der Eigenschaften einer Ressource

[7] Im WebDAV-Standard wird eine Datei oder ein komplettes Verzeichnis als Ressource

Das folgende Beispiel zeigt eine WebDAV-Anfrage zur Ermittlung der Eigenschaften `bigbox`, `author`, `DingALing` und `Random` der Resource `/file` auf dem Rechner `www.foo.bar`:

```
PROPFIND /file HTTP/1.1
Host: www.foo.bar
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:propfind xmlns:D="DAV:">
  <D:prop xmlns:R="http://www.foo.bar/boxschema/">
    <R:bigbox/>
    <R:author/>
    <R:DingALing/>
    <R:Random/>
  </D:prop>
</D:propfind>
```

Als Antwort auf diese Anfrage könnte zum Beispiel folgendes geliefert werden:

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://www.foo.bar/file>
    <D:propstat>
      <D:prop xmlns:R="http://www.foo.bar/boxschema/">
        <R:bigbox>
          <R:BoxType>Box type A</R:BoxType>
        </R:bigbox>
        <R:author>
          <R:Name>J.J. Johnson</R:Name>
        </R:author>
      </D:prop>
```

bezeichnet.

```
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
    [...]
  </D:response>
  <D:responsedescription> [...]
  </D:responsedescription>
</D:multistatus>
```

Die Erweiterung um neue HTTP-Methoden kann zu Problemen führen, wenn die Anfrage über einen Proxy an den Webserver weitergeleitet wird. Ein Proxy ist unter Umständen mit den Erweiterungen nicht vertraut und weigert sich, nicht HTTP-konforme Anfragen durchzureichen und meldet stattdessen einen Fehler an der Webbrowser zurück.

Durch diese Art der Protokollerweiterung ist es aber möglich, auf ein und dieselbe URL-Adresse mit einem Webbrowser und WebDAV-Client zuzugreifen. Der Browser wird nach Eingabe der URL eine `GET`-Anfrage an den Server schicken, der WebDAV-Client hingegen eine `PROPFIND`-Methode. Auf der Seite des Servers kann dies entsprechend behandelt werden und je nach Client das korrekte Format zurückgeliefert werden.

3.5 Verschlüsselung

3.5.1 Kryptografische Grundlagen

Normale TCP/IP-Verbindungen im Internet übertragen die Steuer- und Nutzdaten im Klartext, das heißt ohne Verschlüsselung. Dadurch lassen sich sensible Daten, wie zum Beispiel Passwörter, an jedem Netzknoten oder an Verbindungen zwischen diesen abhören, mit deren Hilfe man dann unberechtigt Zugang zu Systemen erlangen kann.

Als Abhilfe wurden schon früh Verschlüsselungsalgorithmen entwickelt, bei denen Sender und Empfänger dasselbe Verschlüsselungspasswort verwenden. Diese symmetrischen Verfahren haben jedoch den Nachteil, dass beide Parteien sich zuvor über einen sicheren Kanal auf ein Passwort verständigen müssen. Um ungewolltes Abhören zu vermeiden, wird für jeden Partner ein eigener Schlüssel benötigt. Bei der Anzahl der Rechner, die über das Internet miteinander kommunizieren, ist dies nicht realisierbar.

Daher wurden schon in den siebziger Jahren asymmetrische Verschlüsselungsverfahren entwickelt. Hierbei hat jede Partei einen geheimen privaten Schlüssel und einen zugehörigen öffentlichen Schlüssel. Daten, die mit einem öffentlichen Schlüssel kodiert wurden, können nur mit dem zugehörigen privaten Schlüssel wieder entschlüsselt werden[8]. So kann ein Rechner, der mit einem anderen kommunizieren will, einfach den öffentlichen Schlüssel dieses Rechners anfordern und seine Daten mit diesem verschlüsseln.

In der Praxis wird typischerweise aus Geschwindigkeitsgründen eine Kombination aus beiden Verfahren eingesetzt: Zuerst wird mit einem asymmetrischen Verfahren ein so genannter Verbindungsschlüssel übertragen, mit dessen Hilfe dann die restliche Kommunikation symmetrisch verschlüsselt wird.

Eine detaillierte Beschreibung der gebräuchlichen Verschlüsselungsverfahren und ihrer mathematischen Grundlagen findet sich in [Schneier].

3.5.2 Kryptografische Authentifizierung

Ein asymmetrisches Schlüsselpaar lässt sich nicht nur zur Verschlüsselung, sondern auch für digitale Unterschriften nutzen. Dazu werden diese beiden Schlüssel einfach vertauscht: Der geheime private Schlüssel wird zur Kodierung verwendet und der öffentliche Schlüssel zur Dekodierung. Wenn der übertragene Text erfolgreich entschlüsselt werden konnte, ist dies ein Anzeichen dafür, dass er nur vom Inhaber des privaten Schlüssels stammen kann, denn nur er hat Zugriff auf diesen.

Das einzige Problem dabei ist, sicherzustellen, dass ein Schlüsselpaar wirklich dem angegebenen Teilnehmer gehört. Jeder Nutzer kann ein Schlüsselpaar mit einer gefälschten Identität erstellen und den öffentlichen Schlüssel bekanntgeben, um so Zugriff auf nicht für ihn bestimmte Daten zu erlangen. Um die Zugehörigkeit eines Schlüsselpaares zu einer Person oder Organisation zu gewährleisten, muss dieses zertifiziert sein.

Eine Zertifizierung ist im Grunde nichts anderes als eine digitale Unterschrift einer unabhängigen dritten Stelle für einen öffentlichen Schlüssel. Auf diese Weise kann man, wenn man der Zertifizierungsstelle vertraut, die Authentizität der Gegenstelle überprüfen. Wenn dies beide Kommunikationspartner durchführen, kann jeder von ihnen sicher sein, mit der gewünschten Gegenstelle zu kommunizieren.

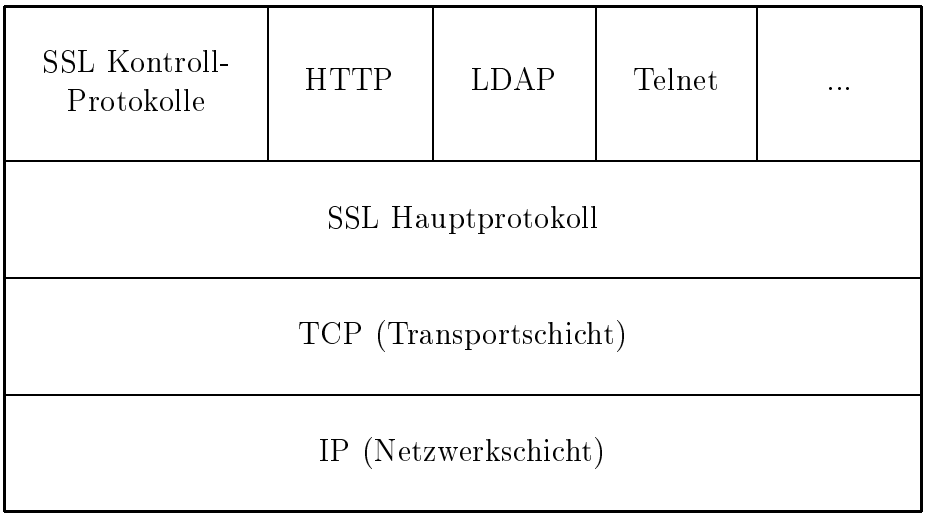
[8] Auch der umgekehrte Weg ist möglich, dies wird im nächsten Unterkapitel besprochen.

3.5.3 Secure Sockets Layers (SSL)

Das SSL-Protokoll, wie unter [SSL 3.0] definiert, wurde von Netscape entwickelt und in der Version 2.0 zum ersten Mal in ihrem Browser Navigator realisiert. Als Verschlüsselungsverfahren wurden RSA (asymmetrisch) und DES (symmetrisch) verwendet[9], in der späteren Version 3.0 kamen dann weitere Verschlüsselungsverfahren sowie die Authentifizierung anhand mehrerer hintereinander angewendeter Zertifikate hinzu.

Jeder Webbrowser, der SSL unterstützt, hat eine Datenbank mit Zertifikaten nach dem X.509-Standard[10], die vom Benutzer auch verändert werden kann. Dadurch ist es möglich, für einen SSL-Webserver selber ein Zertifikat auszustellen anstatt auf die teilweise sehr teuren Zertifikate von internationalen Zertifizierungsstellen zurückzugreifen, die zudem – natürlich gegen Gebühr – jährlich erneuert werden müssen.

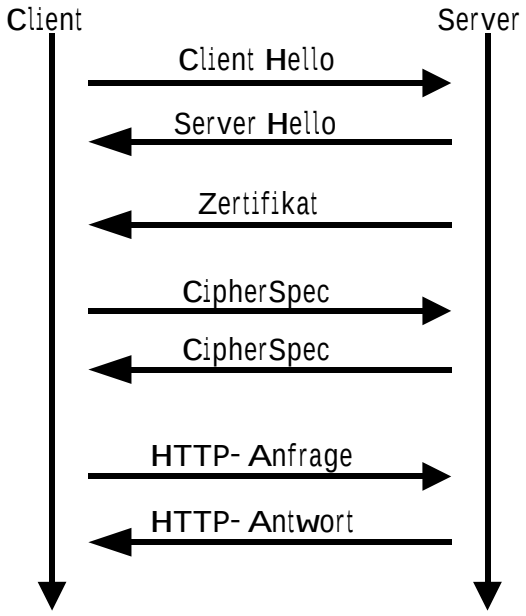
Angesiedelt ist SSL im ISO-Schichtenmodell zwischen der Transport- und Anwendungsschicht, kann also im Prinzip für alle Anwendungen, die auf TCP/IP basieren, verwendet werden. Die folgende Abbildung verdeutlicht dies:



Am häufigsten wird es im Internet im Zusammenhang mit dem HTTP-Protokoll verwendet, die URL der Dokumente beginnt dann mit `https` statt mit `http`. Der Verbindungsaufbau und die Datenübertragung läuft insgesamt

[9] Details zu diesen Verfahren finden sich im Buch von [Schneier].
[10] Siehe [X.509]

nach dem folgenden zeitlichen Schema ab, wobei die Daten nach der Einigung auf ein Verschlüsselungsverfahren in der Phase `CipherSpec` verschlüsslet werden:



Wie man leicht erkennen kann, läuft das komplette HTTP-Protokoll innerhalb einer verschlüsselten Verbindung ab, also auch die Übertragung der Authentifizierungsdaten, die bei der basic-Methode sonst leicht abhörbar wären.

Allerdings hat dies auch zur Folge, dass man beim Einsatz von einem Webserver für mehrere Domains[11] eine eigene IP-Adresse pro Domain benötigt. Die Information, auf welche Domain zugeriffen werden soll, wird erst mit einer HTTP-Anfrage in einem `Host:`-Header übertragen, müsste aber schon zu Beginn des SSL-Protokolls vorliegen.

In der Zwischenzeit ist der Entwurf von SSL Version 3.0 als Internet-Standard ausgelaufen, das heißt SSL wurde nie zu einem formalen Standard. Ein neuer Standard, der auf SSL basiert, namens TLS (Transport Layer Security) Version 1.0 wurde als [RFC 2246] mit geringen Erweiterungen eingereicht und steht kurz vor seiner Verabschiedung.

[11] Dieses Domains werden auch als „virtuelle Hosts" bezeichnet.

3.5.4 SSL-Verbindungen über HTTP-Proxys

Wie aus dem vorherigen Ablaufdiagramm ersichtlich ist, erfordern SSL-Verbindungen schon einen Datenaustausch bevor das eigentliche HTTP-Protokoll zum Einsatz kommt. Daher können solche Verbindungen nicht einfach über einen HTTP-Proxy geleitet werden, sondern müssen speziell behandelt werden.

Um dies zu erreichen wurde eine neue HTTP-Methode `CONNECT` eingeführt, die direkt eine TCP/IP-Verbindung zum Zielrechner aufbaut. Diese Methode ist nicht Bestandteil der HTTP-Spezifikation und wird nur bei der Kommunikation eines Webbrowsers mit einem Proxy verwendet.

Der Proxy leitet nach dem Verbindungsaufbau an alle Daten transparent weiter, das heißt, das SSL-Protokoll läuft direkt zwischen dem Webbrowser auf dem Client und dem Webserver auf den Zielrechner ab. Der Proxy und alle anderen Rechner dazwischen sehen nur den verschlüsselten Datenstrom. Man könnte so also auch ein anderes Protokoll anstatt HTTP über einen oder mehrere Proxys durchleiten.

3.6 Lightweight Directory Access Protocol (LDAP)

Der Verzeichnisdienst LDAP [RFC 2251] ging aus dem wesentlich komplexeren X.500-Protokoll hervor, um es auf die hauptsächlich in der Praxis verwendeten Fähigkeiten zu reduzieren. Das X.500-Protokoll, wie unter [ISO 9594] spezifiziert, setzte ursprünglich auf dem OSI-Schichtenmodell als Netzwerkprotokoll auf, dieses wurde jedoch auf Grund der zu langwierigen Spezifikationszeit bis auf ein paar experimentelle Forschungsprojekte nicht eingesetzt.

Als vollwertiger Verzeichnisdienst bietet LDAP weitreichende Möglichkeiten zur Speicherung von beliebigen hierarchischen Datenstrukturen. Besonders die Möglichkeit, in kompletten Unterbäumen nach einem Muster zu suchen macht es zu einem sehr mächtigen Werkzeug, das andere proprietäre Verzeichnisdienste als Standard ablösen kann.

In diesem Projekt wird LDAP jedoch nicht zur Speicherung oder Suche von Informationen verwendet, sondern lediglich zur Authentifizierung von Benutzern. Dabei wird zuerst, ohne sich anzumelden, auf dem LDAP-Server nach der gewünschten Benutzerkennung gesucht. Wenn diese gefunden wur-

de, wird genau dieselbe Anfrage noch einmal wiederholt, diesmal aber mit Anmeldung. Wenn diese Anmeldung erfolgreich war, weiß man, dass der Benutzername mit zugehörigem Kennwort gültig ist. Spezifiziert wird dieser Vorgang in [RFC 2307] und [RFC 2829].

3.7 Common Internet File System (CIFS)

SMB, eine Abkürzung für Server Message Block, ist ein Protokoll, um Verzeichnisse, Drucker, serielle Schnittstellen und weitere Kommunikationsgeräte von anderen Rechnern über ein Netzwerk zugänglich zu machen. Es wurde Anfang der achtziger Jahre von IBM entwickelt, fand seinen Einzug in die PCs aber erst durch die Implementierung von Microsoft in Windows für Workgroups 3.11.

Das Protokoll lässt sich auf fast alle Netzwerkprotokolle, wie z. B. IPX von Novell, DECnet und natürlich auch TCP/IP, aufsetzen. In den meisten Fällen wird hierbei noch das NetBIOS-Protokoll zwischengeschaltet, das dann im Fall von TCP/IP von Microsoft als NBT („NetBIOS over TCP/IP") bezeichnet wird.

In späteren Versionen des SMB-Protokolls trägt im Moment die interne Versionsnummer „NT LM 0.12" und ist in aktuellen Windows-Versionen sowie im frei erhältlichen Samba-Paket[12] implementiert. Sie enthält weitere Dienste wie Namensauflösung und Anmeldung am Netzwerk.

Als große Teile des SMB-Protokolls durch die Samba-Entwickler erforscht und öffentlich dokumentiert wurden, hat sich Microsoft entschlossen, einen Teil ihrer proprietären Erweiterungen ebenfalls offenzulegen und unter dem Namen CIFS (Common Internet File System) [Leach] zu standardisieren.

Durch das Samba-Projekt ist eine Bibliothek für C-Programmierer entstanden, die sehr schnell und einfach die Nutzung aller SMB-Dienste ermöglicht. Mit ihrer Hilfe ist nicht nur der Zugriff auf Verzeichnisse und Drucker möglich, sondern auch die Benutzerauthentifizierung und sogar das Ändern eines Passworts. Leider gibt es zur Zeit keine vollständige Implementierung dieser Client-Funktionalität in Java, es sind aber mindestens zwei Projekte[13] im

[12] Samba ist im Internet unter [Samba] für Unix-Systeme erhältlich.

[13] Ein Projekt wird offiziell vom Samba-Team unterstützt, seine Entwicklung befindet sich jedoch noch im Anfangsstadium. Das zweite Projekt wurde von einer Privatperson gestartet und bietet eine fast vollständige Implementierung, es wurde daher in dieser Arbeit verwendet.

Gange, die genau dieses Ziel verfolgen.

3.8 Extensible Markup Language (XML)

XML, die Extensible Markup Language, ist eine Untermenge des SGML-Standards [ISO 8879], die zur Speicherung und zum Austausch beliebiger hierarchischer Datenstrukturen geeignet ist. Im Prinzip ist es möglich, jedes proprietären Format zur Speicherung von Daten durch XML zu ersetzen, was von einigen Gruppen im Internet auch angestrebt wird.

Wie auch HTML-Dateien, die schon im Abschnitt 3.1 erläutert wurden, sind auch XML-Dokumente reine Textdateien, deren Struktur durch Tags repräsentiert wird. Sie können also auch mit jedem beliebigen Texteditor erzeugt und bearbeitet werden. Außerdem gibt es fertige XML-Bibliotheken für alle gängigen Skript- und Programmiersprachen.

Ein XML-Parser ist in der Lage, die Struktur jeder beliebigen XML-Datei zu analysieren, falls diese ein wohlgeformtes (engl. well formed) XML-Dokument darstellt. Mit ein und derselben Standardkomponente kann also jede Anwendung die hierarchische Anordnung der XML-Tags einlesen, ohne dass immer ein neuer Parser geschrieben werden müsste. Zusätzlich ist es noch möglich, für XML-Dokumente eine Grammatik in Form einer DTD (Document Type Definition) oder eines XML-Schemas (siehe Abschnitt 3.8.3) anzugeben, die dann vom Parser zur Überprüfung der syntaktischen Korrektheit verwendet wird. Ein solches Dokument wird dann als gültig (engl. valid) im Bezug auf diese Grammatik bezeichnet.

3.8.1 Darstellung von HTML-Dokumenten als XML

Im Vergleich zu HTML-Dokumenten müssen XML-Dokumente wesentlich strengeren Richtlinien genügen. Zum Einen spielt die Groß- und Kleinschreibung der Tags eine Rolle, zum Anderen müssen die Werte von Tag-Attributen immer in Anführungszeichen stehen.

Einen weiteren Unterschied gibt es bei den Separator-Tags, also den Tags ohne zugehörige Endmarkierung: Sie müssen in XML-Dokumenten mit einem abschließenden Schrägstrich gekennzeichnet werden. Das HTML-Beispiel aus Abschnitt 3.1 ist kein gültiges XML-Dokument, es kann jedoch leicht zu einem umgewandelt werden. Der Standard für solche Dokumente heißt

XHTML und in [XHTML 1.0] definiert. Er wird von den aktuellen Versionen der gängigen Webbrowser ohne Probleme unterstützt. Als Beispiel für ein XHTML-Dokument folgt nun das modifizierte und gekürzte oben schon angesprochene HTML-Dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
          PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
          "DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="de" lang="de">
  <head>
    <link rel="stylesheet" type="text/css" href="style1.css">
    <title>Einfaches HTML-Beispiel</title>
  </head>

  <body>
    <h1>Überschrift des Beispiels</h1>

    <p>Hier ein Textabsatz.</p>

    <img alt="" height="80" width="80" src="bild1.jpg"/>
  </body>
</html>
```

3.8.2 XSLT

XML-Dokumente bieten neben der Möglichkeit, sie einfach erstellen und einlesen zu können, noch ein standardisiertes Verfahren zur Bearbeitung. Dabei werden wohlgeformte XML-Dokumente in andere wohlgeformte Dokumente oder in ein beliebiges anderes Format umgewandelt. Durch die erste Variante ist es zum Beispiel sehr einfach möglich, nicht kompatible Konfigurationsdateien[14] ineinander zu konvertieren.

Ein anderes Anwendungsbeispiel ist die Erstellung von HTML-Dokumenten aus XML-Vorlagen. So lassen sich eine Art von „Makros" definieren, um zum

[14] Diese müssen zumindest wohlgeformte XML-Dokumente sein.

Beispiel immer wiederkehrende Konstrukte durch einen einzigen Tag mit eventuellen Parametern darzustellen. Dazu werden Formatvorlagen, die mit den Cascading Style Sheets aus Abschnitt 3.2 verwandt sind, unter der Bezeichnung XSL (Extensible Stylesheet Language) eingesetzt. Diese Vorlagen sind selber auch wieder wohlgeformte XML-Dokumente, das heißt sowohl die XML-Schemata als auch die XML-Transformationen werden in XML selber spezifiziert.

Die eigentliche Umwandlung eines XML-Dokuments wird als Transformation bezeichnet. Die Spezifikation dazu lautet XSL Transformations (XSLT) und ist als [XSLT] verfügbar.

Ein Beispiel für die Umwandlung einer Zeitabrechnung, die in einem XML-Format vorliegt, in eine HTML-Datei stellt der folgende Ausschnitt aus einer XSL-Datei vor:

```
<?xml version="1.0"?>
<?cocoon-process type="xslt"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="workdays">
<html>
<head>
  <title>
    Arbeitszeiten von <xsl:value-of select="@name"/>
  </title>
</head>

<body>
  <h1>Arbeitszeiten von <xsl:value-of select="@name"/></h1>
  <xsl:apply-templates>
    <xsl:sort select="workday/@date"/>
  </xsl:apply-templates>
</body>
</html>
</xsl:template>

[...]

<xsl:template match="task">
```

```
  <tr>
    <td><xsl:value-of select="@company"/></td>
    <td><xsl:value-of select="@project"/></td>
    <td><xsl:value-of select="@location"/></td>
    <td align="right"><xsl:value-of select="@duration"/></td>
    <td><xsl:apply-templates/></td>
  </tr>
</xsl:template>
```

3.8.3 XML Schemata

Mit Hilfe von Schemata [XML Schema] lässt sich die erlaubte Struktur einer XML-Datei festlegen. Im Gegensatz zu DTDs (Document Type Definitions), mit deren Hilfe sich die erlaubte Schachtelung von XML-Tags festlegen lässt, gehen Schemata einen Schritt weiter: Sie definieren auch den erlaubten Inhalt, also den Text zwischen Tags, sowie Datentypen von Attributen. So lässt sich zum Beispiel festlegen, dass der Wert eines Attributs `date` ein gültiges Datum, bestehend aus Tag, Monat und Jahr, sein muss.

Neben bereits vordefinierten Datentypen, wie z. B. einer ganzen Zahl oder eines Datums, lassen sich auch weitere komplexe Datentypen aus den schon vorhandenen zusammensetzen. Außerdem besteht bei Zahlen und Datumsangaben die Möglichkeit, einen zulässigen Wertebereich anzugeben.

Ein XML-Parser kann ein Schema zusammen mit einer XML-Eingabedatei verwenden, um deren syntaktische Korrektheit zu garantieren. Das folgende Beispiel stellt einen Ausschnitt eines XML-Schemas zur Definition von digitalen Unterschriften in XML dar:

```
<?xml version='1.0'?>
<!DOCTYPE schema
   SYSTEM 'http://www.w3.org/1999/XMLSchema.dtd'
  [
   <!ENTITY dsig 'http://www.w3.org/2000/02/xmldsig#'>
  ]>


<schema targetNamespace='&dsig;'
```

```
   version='0.1'
   xmlns='http://www.w3.org/1999/XMLSchema'
   xmlns:ds='&dsig;'
   elementFormDefault='qualified'>

<!-- Basic Types Defined for Signatures -->

<simpleType name='CryptoBinary' base='binary'>
    <encoding value='base64'/>
</simpleType>

<!-- Start Signature -->

<element name='Signature'>
  <complexType content='elementOnly'>
    <sequence minOccurs='1' maxOccurs='1'>
      <element ref='ds:SignedInfo' minOccurs='1'
                                   maxOccurs='1'/>
      <element ref='ds:SignatureValue' minOccurs='1'
                                       maxOccurs='1'/>
      <element ref='ds:KeyInfo' minOccurs='0'
                                maxOccurs='1'/>
      <element ref='ds:Object' minOccurs='0'
                               maxOccurs='unbounded'/>
    </sequence>
    <attribute name='Id' type='ID'/>
  </complexType>
</element>

[...]
```

3.9 Debian-Pakete

Bei Unix-Systemen ist es üblich, Software als tar-Archive, die eventuell auch noch komprimiert sein können, auszuliefern. Diese Archive enthalten jedoch außer den eigentlichen Programmdateien keine weiteren Informationen, wie z. B. eine Beschreibung. Daher haben sich in Linux-Distributionen Pakete

zur Verbreitung von Software durchgesetzt, von denen zur Zeit die beiden Systeme `rpm` von RedHat und `dpkg` von Debian am häufigsten im Einsatz sind.

Im Wesentlichen enthalten diese Pakete ein tar-Archiv mit den Programmdaten und ein weiteres Archiv, das die ganzen Steuerinformationen beinhaltet. Zu den Steuerinformationen gehören die Beschreibung, die Abhängigkeiten und Konflikte mit anderen Paketen sowie die Versionsnummer.

3.9.1 Vorteile von Paketen

Mit Hilfe eines oder mehrerer Systemkommandos, die zur Grundausstattung einer Linux-Distribution gehören, ist es möglich, neue Pakete auf einfache Weise zu installieren. Im Gegensatz zu Windows, bei dem jedes Softwareprodukt mit einer eigenen Installationsroutine ausgeliefert wird, sind Linux-Pakete nicht nur kleiner, sondern auch einheitlich zu installieren – und auch wieder einfach und vollständig zu entfernen! Die Vorteile von Paketen im Vergleich zu einfachen tar-Archiven oder proprietären Installationsroutinen sind zusammengefasst:

- Pakete beinhalten sowohl eine Kurzbeschreibung als auch eine ausführliche Beschreibung, die eine Suche in allen verfügbaren Paketen ermöglicht und außerdem eine Klassifikation erlaubt.
- Zwischen Paketen können Abhängigkeiten und Konflikte festgelegt werden, so dass gemeinsame Komponenten mehrerer Pakete in ein weiteres Paket ausgelagert werden können und all diese Pakete eine Abhängigkeit von dem neuen Paket definieren. Bei einer Installation eines Pakets können auf Wunsch auch automatisch alle abhängigen Pakete installiert werden.
- Pakete können Skripts enthalten, die vor und nach der Installation ausgeführt werden. Mit diesen Skripts ist es somit möglich, noch notwendige Konfigurationsarbeiten durchzuführen. Vor dem Entfernen eines Pakets können diese Schritte dann automatisch wieder rückgängig gemacht werden.
- Für jede auf dem System vorhandenen Datei kann festgestellt werden, aus welchem Paket sie stammt. Dadurch kann vermieden werden, dass Dateien überschrieben werden, wenn sie auch in anderen Paketen vorhanden sind. Bei einem Update eines Pakets werden in der neuen Version nicht mehr vorhandene Dateien automatisch gelöscht.

- Bei den Abhängigkeiten und Konflikten können nicht nur real existierende Pakete, sondern auch „virtuelle Pakete" angegeben werden. So kann ein Servlet-Engine angeben, dass sie zum Betrieb einen Webserver - aber ganz egal welchen - benötigt. Alle Pakete, die einen Webserver beinhalten, erklären dies und auch die Tatsache, dass nur ein einziger Webserver installiert sein kann[15], in ihren Steuerinformationen.
- Konfigurationsdateien können in Paketen speziell gekennzeichnet werden. Dadurch werden sie, wenn sie vom Systemadministrator verändert wurden, bei der Installation einer neuen Version nicht überschrieben.

3.9.2 Erstellung von Debian-Paketen

Debian-Pakete können mit Hilfe von Programmen aus dem `dpkg-dev`-Paket erzeugt werden. Hierzu werden alle zu installierenden Dateien in eine Verzeichnishierarchie kopiert, die Konfigurationsdateien markiert und die Skripts zur automatischen Konfiguration hinzugefügt. Bei Open Source-Programmen, also Programmen, deren Quellcode frei verfügbar ist, ist die Erzeugung von Debian-Paketen besonders einfach: In den Verzeichnisbaum des Quellcodes kann einfach ein Unterverzeichnis eingefügt werden, das alle notwendigen Dateien zur Erstellung eines Paket enthält.

Um nun noch die Beschreibung, die Klassifikation, die Abhängigkeiten und Konflikte spezifizieren zu können, ist eine spezielle Steuerdatei (engl. control file) zu erstellen. Diese hat ein vorgegebenes Format und liegt typischerweise unter dem Namen `debian/control` im Hauptverzeichnis vor. Ein Beispiel für eine solche Datei ist die Steuerdatei des `tomcat`-Pakets:

```
Source: tomcat
Section: contrib/web
Priority: optional
Maintainer: Stefan Gybas <sgybas@debian.org>
Build-Depends: debhelper (>= 2.1.0), ant (>= 1.1-2), j2sdk1.3
Standards-Version: 3.2.1

Package: tomcat
Architecture: all
```

[15]Theoretisch könnten auch mehrere Pakete gleichzeitig installiert sein, die einen Webserver beinhalten. Diese Webserver dürfen dann jedoch nicht gleichzeitig versuchen, TCP/IP-Verbindungen auf dem Standard-Port 80 anzunehmen.

```
Section: contrib/web
Priority: optional
Depends: jdk1.1-dev | ibm-jdk1.1-installer | j2sdk1.3,
         libxerces-java, libservlet2.2-java
Description: Java Servlet 2.2 engine with JSP 1.1 support
 Jakarta-Tomcat is the reference implementation for the Java
 Servlet 2.2 and JavaServer Pages (JSP) 1.1 specification from
 the Apache Jakarta project.
 .
 [Weitere Beschreibung]
```

Außerdem ist eine Steuerdatei für das Standardprogramm `make` zu erstellen, die dann die eigentliche Arbeit erledigt. Hierzu kann auf eine große Auswahl an Hilfsprogrammen zurückgegriffen werden, es müssen lediglich die `make`-Ziele `build`, `clean`, `install`, `binary-arch` und `binary-indep` vorhanden sein.

Das folgende Beispiel zeigt einen Ausschnitt aus dieser Datei, die unter dem Namen `debian/rules` vorhanden sein muss. Hier werden Hilfsprogramme aus dem Paket `debhelper` verwendet, die alle mit dem Präfix `dh_` beginnen:

```
#!/usr/bin/make -f
# debian/rules file for tomcat (uses debhelper V2)

[...]

build: build-stamp
build-stamp:
dh_testdir
# call ant to build Tomcat in the build/ directory
$(JAVA) -Dant.home=/usr/share/ant org.apache.tools.ant.Main
touch build-stamp

install: build
dh_testdir
dh_testroot
dh_clean -k
dh_installdirs

# Build the Tomcat JAR
```

```
cd build/classes && $(JAR) cf \
             ../../debian/tomcat/usr/share/java/tomcat.jar org

[...]
```

Eine ausführliche Beschreibung zur Erstellung von Debian-Paketen kann in der offiziellen Dokumentation von [Jackson] nachgelesen werden. Dort finden sich auch Hinweise auf Beispiele, die einen Einstieg in diese Materie vereinfachen.

3.10 Java-Standards

3.10.1 Java Beans

Mit der Java Version 1.1 wurde von Sun ein Komponentenmodell entwickelt, mit dessen Hilfe ein komplexes Softwareprojekt aus einfachen Bausteinen zusammengesetzt werden kann. Solche Komponenten sollen außerdem die Wiederverwendbarkeit von Code erhöhen, um dadurch Entwicklungskosten und -zeit einzusparen.

Standardisiert wurde dieses Verfahren unter dem Namen Java Beans. Die zur Zeit aktuelle Version 1.01 der Spezifikation ist unter [Hamilton] veröffentlicht.

Bei Java Beans handelt es sich um normale Java-Klassen, die jedoch einige zusätzliche Eigenschaften erfüllen müssen. Durch sie wird es möglich, Beans auch mit Hilfe von grafischen Entwicklungsumgebungen miteinander zu verknüpfen, ohne dass eine Zeile Code geschrieben werden muss. Die Attribute einer Klasse, also deren Instanz- und Klassenvariablen, werden bei Java Beans „Eigenschaften" (engl. properties) genannt. Ansonsten kann man sie aber wie in jeder normalen Java-Klasse auch verwenden.

Die zusätzlichen Eigenschaften für Java Beans sind im Einzelnen:

- Jede Bean-Klasse muss einen öffentlichen Konstruktor besitzen, der keine Parameter erwartet. Nur so ist es möglich, eine Instanz dieser Klasse zu erzeugen, ohne ihre genauen Eigenschaften zu kennen.
- Die Bean-Klasse muss serialisierbar sein, das heißt sie muss das Interface `Serializable` aus dem Paket `java.io` implementieren.

- Der Aufbau der Klasse, insbesondere die Namen der Methoden, müssen den Design-Patterns der Java Beans-Spezifikation genügen. Methoden zum Lesen und Setzen von Eigenschaften müssen zum Beispiel mit `get` bzw. `set` beginnen.

Ein einfaches Beispiel für ein Bean mit nur einem einzigen Attribut und den zugehörigen Methoden zur Manipulation dieses Attributs ist folgende Klasse:

```
import java.io.Serializable;

public class Person implements Serializable {

    // Attribute bzw. Properties

    private String name;

    // Konstruktoren

    public Person() {
    }

    public Person(String name) {
        this.name = name;
    }

    // Methoden zum Zugriff auf die Attribute

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

Die Kommunikation zwischen einzelnen Beans geschieht im Wesentlichen über Ereignisse (engl. events), wie es auch schon bei den grafischen Oberflächen AWT und Swing der Fall ist. Alle Beans, die an Ereignissen in einer

anderen Bean interessiert sind, registrieren sich bei dieser als so genannte `EventListener`. Bei ihnen wird dann beim Eintreten dieses Ereignisses eine Methode aufgerufen.

Im Gegensatz zu Java Beans widmen sich die Enterprise Java Beans (EJB) der Kommunikation zwischen Prozessen. Hierzu setzen sie auf bewährte Standards, wie zum Beispiel CORBA, auf.

3.10.2 Java Servlets

Webserver bieten schon lange die Möglichkeit, Programme zur Erzeugung von dynamischen Seiten ablaufen zulassen. Sie verwendeten bisher hauptsächlich das Common Gateway Interface (CGI)[16], das alle Daten zu einer HTTP-Anfrage in Umgebungsvariablen übergibt. Dadurch konnte CGI mit allen Programmiersprachen verwendet werden, hauptsächlich wurden aber Perl und C eingesetzt.

Weitere übermittelte Daten in der Anfrage, vor allem Formularinhalte, können von CGI-Programmen von der Standard-Eingabe gelesen werden. Die dynamisch erstellte Seite wird einfach an die Standard-Ausgabe geschickt und vom Webserver direkt an den Browser weitergeleitet. Dazu ist es notwendig, ein Dokument zu erstellen, das ein Webbrowser anzeigen kann, also beispielsweise eine HTML-Seite oder ein GIF-Bild.

Die Art der Parameterübergabe in Umgebungsvariablen widersprach jedoch dem Gedanken der Objektorientierung. Deshalb wurde von Sun ein Standard für Java-Programme geschaffen, der als „Servlet" bekannt ist. Die Spezifikation selber ist unter [Servlet 2.2] in der aktuellen Version 2.2 erhältlich, mittlerweile sind zu diesem Themenbereich aber auch zahlreiche Bücher erschienen, wie zum Beispiel [Hunter].

Für jede HTTP-Anfrage, die ein Webserver erhält und an ein Servlet weiterleitet, wird eine `service()`-Methode aufgerufen. Jede Servlet-Klasse muss diese Methode besitzen, da sie das Interface `javax.servlet.http.HttpServlet` implementiert. Dieser Methode werden in zwei Parametern die notwendigen Daten zur Anfrage und zur Antwort mitgeteilt. Die Anfrage wird in einer Instanz der Klasse `javax.servlet.http.HttpRequest` übergeben, die wiederum Methoden zur Abfrage der HTTP-Methode, des gewünschten Dokuments oder der Kopfzeilen bietet.

[16] [CGI 1.1] enthält die CGI-Spezifikation in der Version 1.1

Um die Zugriffe eines Benutzers über mehrere HTTP-Anfragen hinweg verfolgen zu können, werden von der Servlet-Engine so genannte Sitzungen (engl. sessions) zur Verfügung gestellt. Damit können Attribute eines Benutzers, bei einer Online-Bestellung zum Beispiel die im Warenkorb befindlichen Produkte, lokal gespeichert und bei der nächsten Anfrage wieder gelesen werden. Technisch wird dies durch Cookies[17] realisiert, die eine eindeutige Kennzeichnung einer Sitzung auf dem Webbrowser speichern und bei jeder HTTP-Anfrage wieder an den Webserver zurückschicken.

Wie schon in Kapitel 3.4 erwähnt, müssen die Antworten für Webbrowser und WebDAV-Clients in unterschiedlichen Formaten generiert werden. Deshalb ist es speziell in diesem Projekt notwendig nach der HTTP-Methode zu unterscheiden. Ein Beispiel für ein solches Servlet befindet sich in Abschnitt 4.2.

3.10.3 JavaServer Pages (JSP)

Bei Webseiten, die von CGI-Programmen oder von Servlets erzeugt werden, ist meist ein großer Teil statisch vorgegeben, zum Beispiel der Kopf mit einer Überschrift und einer einführenden Erläuterung. Nur ein kleiner Teil dieser Seite wird abhängig von der aktuellen Anfrage erzeugt, meist durch Abfragen einer Datenbank.

Typischerweise werden die statischen Teile einer Seite von einem Grafiker erstellt. Dieser HTML-Code muss dann von dem CGI-Programm oder von dem Servlet jeweils zusätzlich mit ausgegeben werden. Realisiert wird dies, indem der HTML-Code Zeile für Zeile durch einzelne `print()`-Anweisungen ausgegeben wird. Wenn nun vom Grafiker Änderungen am Layout vorgenommen werden, müssen diese von Hand im Programmcode nachgeführt werden. Dies ist meist sehr schwer, da die HTML-Struktur innerhalb von Programmcode nur mit grossem Aufwand übersichtlich dargestellt werden kann.

Um genau diesem Problem abzuhelfen, wurde eine Möglichkeit geschaffen, direkt im HTML-Code einzelne Programmanweisungen einzufügen. Dazu wurden neue HTML-Tags definiert, so dass es sich noch immer um eine gültige HTML-Seite handelt, die auch mit grafischen Hilfsprogrammen bearbeitet werden kann. Solche Lösungen existieren für Perl[18] und andere Programmiersprachen, im Fall von Java Servlets sind JSP-Dateien, wie unter [JSP 1.1]

[17] Cookies sind in der HTTP-Spezifikation [RFC 2616] erläutert.

[18] Eine davon ist zum Beispiel das Modul `HTML::EP`, das auf jedem Perl-Mirror zu finden ist.

spezifiziert, die standardisierte Methode.

Das eigentliche Problem, nämlich eine saubere Trennung zwischen HTML-Struktur und Programmlogik, ist damit noch nicht gelöst. Ob nun HTML-Code innerhalb eines Programms oder Programmfragmente innerhalb einer HTML-Datei verwendet werden, spielt bei komplexen Dokumenten keine Rolle. In beiden Fällen entsteht eine Datei, die praktisch nicht wartbar ist.

Um diese Trennung dennoch zu erreichen, kann man in den JSP-Dateien ausschließlich Java Beans und einfache Kontrollstrukturen, wie zum Beispiel `for`- oder `if`-Anweisungen, einsetzen. Der größte Teil der Programmlogik ist hier in den Methoden der Beans enthalten und dadurch vollständig vom HTML-Code getrennt.

Das folgende Beispiel demonstriert die Einbindung eines Java Beans in eine JSP-Seite und den Zugriff auf Eigenschaften dieses Beans. Außerdem wird noch Java-Code direkt innerhalb der Tags `<%` und `%>` eingefügt:

```
<html>
<head>
    <title>JSP-Beispiel</title>
</head>

<body>
    <jsp:useBean id='clock' scope='page'
                 class='JspCalendar' type="JspCalendar" />

    Day of month: <jsp:getProperty name="clock"
                                   property="dayOfMonth"/>
    Year: <jsp:getProperty name="clock" property="year"/>

    [...]

    <% for(int i=1; i<=5; i++) { %>

        <p>Dies ist Textzeile <%= i %></p>

    <% } %>

</body>
</html>
```

3.10.4 XML mit Java

XML-Dokumente können von allen Programmiersprachen aus über zwei einheitliche Schnittstellen bearbeitet werden. Von den meisten erhältlichen XML-Parsern werden beide Schnittstellen unterstützt, so dass je nach Bedarf die günstigere gewählt werden kann.

1. Die SAX-Schnittstelle ist, wie ihr ausgeschriebener Name „Simple API for XML" schon andeutet, die einfachere der beiden Varianten. Sie unterstützt nur das Einlesen von XML-Dokumenten und erzeugt bei jedem gefundenen Tag ein Ereignis, das dann individuell behandelt werden kann. Hierbei wird die XML-Quelle sequenziell eingelesen und nicht komplett im Hauptspeicher gehalten. Dadurch eignet sich SAX auch für sehr große XML-Dokumente.

2. Die zweite Schnittstelle ist DOM, das „Document Object Model", bei der ein XML-Dokument komplett in den Arbeitsspeicher geladen und dort in seine Baumstruktur zerlegt wird. Sie ermöglicht im Gegenzug aber die direkte Manipulation der Baumstruktur, so können auf einfache Weise neue Unterbäume eingefügt oder verschoben werden. Durch einen einfachen Funktionsaufruf kann das geänderte XML-Dokument wieder gespeichert werden.

   Der Zugriff und die Manipulation von DOM-Bäumen ist sehr zeitaufwändig. Deshalb werden alle benötigten Daten zur Laufzeit in normalen Java-Objekten gespeichert. Besonders beim Einsatz von Hash-Tabellen ist dieser Weg deutlich schneller.

SAX und DOM sind beides universelle Schnittstellen[19], die auch in Skriptsprachen verwendet werden können. Deshalb ist ihr Aufbau nicht objektorientiert, im Besonderen gibt es keine einfache Möglichkeit, ein Objekt in einem XML-Baum umzuwandeln. Hierbei könnten alle Attribute eines Objekts in einem Unterbaum rekursiv gespeichert werden, was im Prinzip der in Java schon vorhandenen Serialisierung entspricht [20].

Da die Umwandlung von Objekten in XML-Dateien selbst programmiert werden muss, würde es keine Vorteile bringen, wenn DOM verwendet wird. Daher wird hier SAX eingesetzt und beim Einlesen der XML-Datei einfach deren Inhalt in Java-Objekte kopiert. Die genaue Realisierung dieses Verfahrens zusammen mit ein Beispiel befindet sich im Kapitel 4.4.

[19] Ihre Spezifikationen sind unter [SAX 2.0] und [DOM 1.0] verfügbar.

[20] Es gibt einige Ansätze, um genau dies zu realisieren, diese sind aber noch in einem experimentellen Stadium und wurden deshalb hier nicht verwendet.

3.10.5 Web-Anwendungen

Im Allgemeinen muss zur Installation von Anwendungen, die über eine Web-Oberfläche gesteuert werden, die Konfiguration des Webservers abgeändert werden. Typischerweise müssen dazu bestimmte Dateien oder Verzeichnisse mit einem Skript oder Programm verknüpft werden. Dieses Programm erzeugt dann je nach aufgerufener Datei eine dynamische HTML-Seite.

Bei traditionellen Anwendungen, die auf einer Workstation installiert werden und lokal eine grafische Oberfläche bieten, fallen zwar ähnlich komplexe Konfigurationsschritte an, diese stellen aber heutzutage durch langjährige Erfahrung kein Problem mehr dar. So kann während der Installation ein Eintrag in der Windows-Registry problemlos hinzugefügt, bearbeitet oder gelöscht werden.

Im Vergleich dazu gibt es webbasierte Anwendungen erst seit wenigen Jahren. Bisher gab es hier noch keine Technologie, die eine Installation von neuen Anwendungen auf beliebigen Webservern ermöglichte. Je nach Hersteller und Typ des Webservers waren unterschiedlich aufwändige Schritte nötig, um selbst ein einfaches Verzeichnis mit einer Anwendung zu verknüpfen.

Für Web-Anwendungen, die mit Hilfe von Java Servlets ab der Version 2.2 geschrieben wurden, existiert nun aber so eine Möglichkeit. Jeder Webserver, der der aktuellen Servlet-Spezifikation genügt, ist in der Lage, neue Anwendungen automatisch einzubinden und zu konfigurieren. Dazu muss die Anwendung nur in einem Web-Archiv vorliegen und in das Anwendungsverzeichnis des Webservers kopiert werden.

Für Java-Anwendungen gab es bisher schon die Möglichkeit, sie in einem Archiv weiterzugeben. Diese so genannten JAR-Archive sind mit den unter Windows populären ZIP-Archiven verwandt: Außer den Java-Klassen enthalten sie noch ein spezielles Verzeichnis `META-INF`, das Steuerinformationen beinhaltet. Vor allem kann hier die Hauptklasse festgelegt werden, die beim Start einer Applikation aufgerufen wird. Mit dieser Technik ist es seit der Java-Version JDK 1.1 möglich, Anwendungen direkt aus JAR-Archiven heraus zu starten, ohne sie vorher zu entpacken.

Web-Archive tragen typischerweise die Dateiänderungen `.war` und stellen eine Erweiterung der JAR-Archive dar. Sie enthalten außer dem `META-INF`-Verzeichnis noch ein zweites Verzeichnis `WEB-INF`, das unter anderem eine Konfigurationsdatei für den Webserver enthält. Im Gegensatz zu JAR-Archiven werden die Klassendateien nicht im Hauptverzeichnis, sondern auch in einem Unterverzeichnis von `WEB-INF` abgelegt. Ein Web-Archiv hat damit

diesen Aufbau:

`/`	Hauptverzeichnis und Unterverzeichnisse für den direkten Zugriff über den Webserver. Hier können beliebige Dateien abgelegt werden, die einfach an den Client geschickt werden, falls ihre Endung nicht mit einem Servlet in `web.xml` verknüpft wurde.
`/META-INF/*`	Informationen über den Archiv-Inhalt (wie schon bei JAR-Archiven)
`/WEB-INF/classes/`	entpackte Klassendateien mit ihrer vollständigen Paketstruktur. Dieses Verzeichnis ist im Klassenpfad der jeweiligen Anwendung und wird automatisch aktualisiert, wenn sich eine Klasse in diesem Verzeichnis ändert.
`/WEB-INF/lib/`	HAR-Archive, die in den Klassenpfad der Web-Anwendung eingefügt werden. Dadurch kann jede Anwendung Klassen unter dem gleichen Namen enthalten, die sich gegenseitig nicht beeinflussen. Dies ist zum Beispiel sinnvoll, wenn unterschiedliche Versionen einer Klassenbibliothek verwendet werden müssen.
`/WEB-INF/web.xml`	Konfigurationsdatei für die Servlet-Engine

Die Konfigurationsdatei für den Webserver im `WEB-INF`-Verzeichnis ist, wie die Dateiendung schon vermuten lässt, eine XML-Datei. In ihr lassen sich alle Einstellungen, die global für den ganzen Webserver konfiguriert werden können, für die betreffende Web-Anwendung überschreiben. Ein kleiner Ausschnitt aus einer solchen Datei demonstriert die Regelung von Zugriffsrechten:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>
    <security-constraint>
      <web-resource-collection>
         <web-resource-name>Protected Area</web-resource-name>
         <url-pattern>/webdav/*</url-pattern>
         <http-method>DELETE</http-method>
         <http-method>GET</http-method>
         [...]
      </web-resource-collection>
      <auth-constraint>
         <role-name>admin</role-name>
      </auth-constraint>
    </security-constraint>

    <login-config>
      <auth-method>BASIC</auth-method>
      <realm-name>Admin</realm-name>
    </login-config>
</web-app>
```

Wenn ein Web-Archiv in das Anwendungsverzeichnis eines Webservers kopiert wird, steht diese Anwendung sofort unter dem Namen des Archivs zur Verfügung. Wenn also die Datei `webdav.jar` installiert wird, kann die darin enthaltene Anwendung auf dem Webserver unter dem Pfad `/webdav/` genutzt werden. Für diesen Pfad und alle darunter liegenden Verzeichnisse gelten dann die Konfigurationseinstellungen aus der Datei `WEB-INF/web.xml` im Web-Archiv.

Kapitel 4

Implementierung und Test

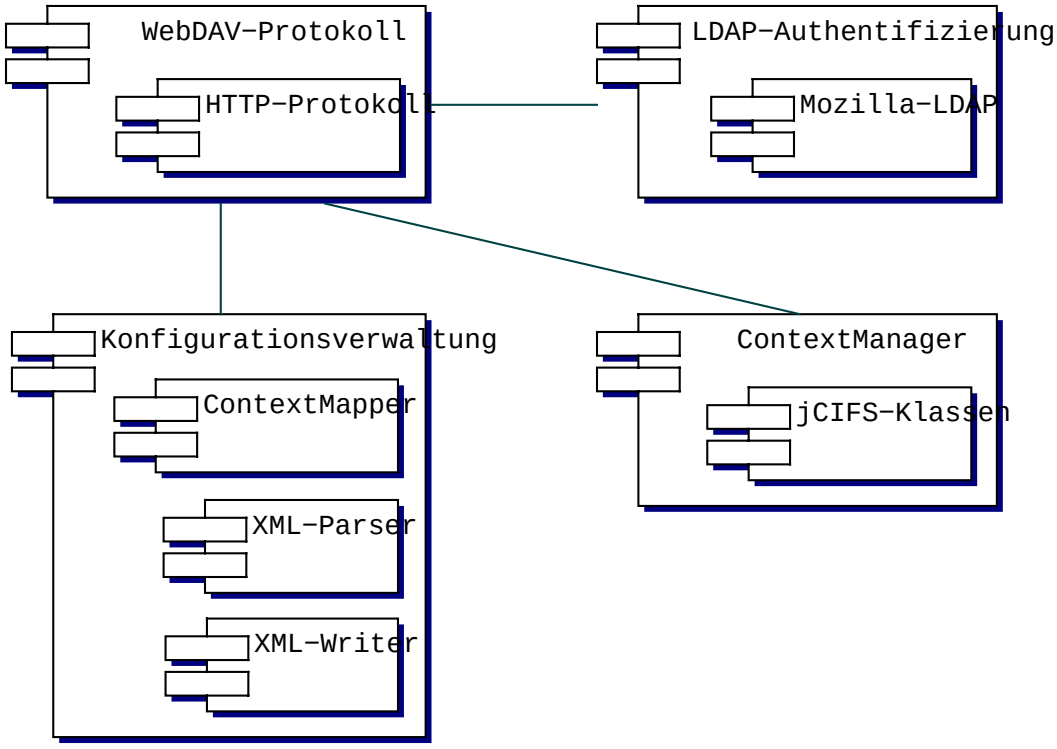
4.1 Überblick

Die Grundstruktur des Komplettsystems wurde schon in Abschnitt 2.2 erläutert. Nun wird das WebDAV-Servlet, das in dem Diagramm in Abschnitt 2.2 rechts dargestellt ist, näher besprochen. Die dabei eingesetzten Protokolle und Technologien sind im vorherigen Kapitel allgemein erklärt, ihre Funktionsweise wird daher hier als bekannt vorausgesetzt.

Etwas detaillierter dargestellt hat das WebDAV-Servlet vier Komponenten, die alle über die Hauptkomponente, die selber das WebDAV- und HTTP-Protokoll implementiert, miteinander kommunizieren.

Die folgenden Abschnitte widmen sich diesen Komponenten, gefolgt von einem Abschnitt über die Aufteilung dieses Systems in Debian-Pakete und in Web-Archive und schließlich beschäftigt sich ein letzter Abschnitt vor der Zusammenfassung mit Testläufen.

Den genaue Aufbau, zusammen mit den jeweils vorhandenen Unterkomponenten, zeigt das nächste UML-Diagramm:



4.2 Das WebDAV-Servlet

Das zu erstellende Servlet soll ausschließlich die Methoden der Servlet-Spezifikation einsetzen. Dadurch ist sichergestellt, dass es mit jeder Servlet-Engine zusammenarbeitet. Trotzdem muss vor Beginn der Implementierung entschieden werden, welche Servlets-Engine vor allem für Tests verwendet werden soll. Hierbei fiel die Wahl auf Tomcat vom Apache/Jakarta-Projekt, das im Internet frei unter [Tomcat] erhältlich ist, da es die von Sun unterstützte Referenz-Implementierung darstellt.

Für die Realisierung der Servlet-Klassen wurde zum Teil auf Beispiele der nächsten Tomcat-Generation[1] zurückgegriffen. Dort befinden sich im Paket `org.apache.catalina.servlets` eine Klasse `DefaultServlet`, die die

[1] Diese Version 4.0 ist zur Zeit noch in einem Anfangsstadium, daher wird ihr Einsatz in der Praxis nicht empfohlen. Die übernommenen Code-Fragmente funktionieren aber auch einwandfrei mit älteren Versionen oder anderen Servlet-Engines.

HTTP-Methoden `GET` und `POST` bearbeiten kann.

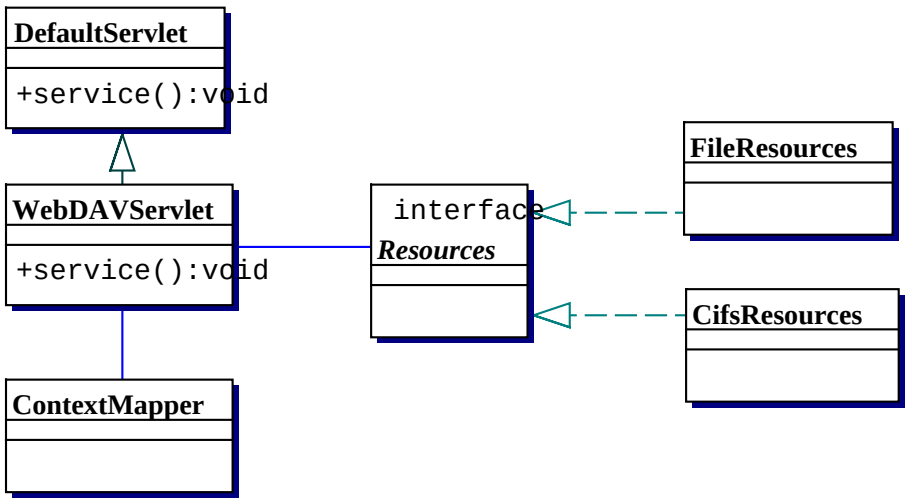
Falls ein `GET`-Zugriff auf ein Verzeichnis erfolgt, wird dessen Inhalt mit Hilfe der Klasse `DirectoryBean` dargestellt. Diese Ansicht wurde erweitert, um das Hochladen und Löschen von Dateien innerhalb eines HTML-Formulars zu ermöglichen.

Zur oben genannten Klasse `DefaultServlet` gibt es noch eine Unterklasse `WebDAVServlet`, die die erweiterten WebDAV-Methoden bearbeitet. Diese beiden Klassen greifen jedoch nicht direkt auf ein lokales Verzeichnis zu, sondern überlassen diese Arbeit einer anderen Klasse. Sie muss vom `Resources`-Interface die folgenden Zugriffsoperation implementieren, die dort auch ausführlich kommentiert sind:

```
interface Resources {
  String getMimeType(String file);
  String getRealPath(String path);
  URL getResource(String path);
  InputStream getResourceAsStream(String path);
  boolean exists(String path);
  long getResourceModified(String path);
  long getResourceCreated(String path);
  long getResourceLength(String path);
  boolean isCollection(String path);
  String[] getCollectionMembers(String path);
  boolean setResource(String path, InputStream content);
  boolean createCollection(String path);
  boolean deleteResource(String path);
}
```

Bisher wurden von den Tomcat-Autoren zwei solcher Klassen implementiert: `FileResources` erlaubt den Zugriff auf lokale Verzeichnisse und `JarResources` den Zugriff auf Dateien innerhalb von JAR-Archiven. Welches lokale Verzeichnis bzw. welches JAR-Archiv für eine bestimmte URL verwendet werden soll, kann in der globalen Konfigurationsdatei festgelegt werden. Dies ist jedoch so eng mit der Implementierung des Tomcat-Kerns verbunden, dass dieses Zuordnungsverfahren hier nicht verwendet werden kann. Stattdessen wird die eigene Klasse `ContextMapper` eingesetzt, die die Zuordnung anhand der Konfigurationsdatei durchführt. Dieses Konfigurationsschema wird im übernächsten Abschnitt erläutert.

Damit ergibt sich folgendes Zusammenspiel der Klassen:



Jede Anfrage eines Browsers oder eines WebDAV-Clients an die Anwendung wird an das WebDAV-Servlet weitergeleitet. Dies wird durch einen entsprechenden Eintrag in der Konfigurationsdatei für die Web-Anwendung[2] sichergestellt. Das Servlet ermittelt nun aus dieser Anfrage die HTTP-Methode und die angeforderte Datei.

Falls es sich um eine reine HTTP-Anfrage handelt, also die Methode `GET` oder `POST` verwendet wurde, wird die Anfrage an die Oberklasse `DefaultServlet` weitergeleitet. Im anderen Fall wird sie direkt in der `WebDAVServlet`-Klasse abgearbeitet und je nach Methode an ein Unterprogramm verwiesen. Dazu wird die `service()`-Methode der Klasse `HTTPServlet` überschrieben, um zuerst auch noch die Authentifizierung durchzuführen:

```
protected void service(HttpServletRequest req,
                       HttpServletResponse resp)
    throws ServletException, IOException {

    String method = req.getMethod();

    // Die richtige Funktion für jede HTTP-Methode
    // auswählen und vorher die Authentifizierung prüfen
```

[2] Dies ist die Datei `web.xml` im Verzeichnis `WEB-INF`. Details hierzu finden sich in Abschnitt 3.10.5.

```
    if (method.equals(METHOD_PROPFIND)) {
        if (checkAuth(req, resp)) doPropfind(req, resp);
    } else if (method.equals(METHOD_PROPPATCH)) {
        if (checkAuth(req, resp)) doProppatch(req, resp);
    } else if (method.equals(METHOD_MKCOL)) {
        if (checkAuth(req, resp)) doMkcol(req, resp);
    [...]
    } else {
        // Methoden, die keine WebDAV-Erweiterungen sind,
        // werden von DefaultServlet bearbeitet
        super.service(req, resp);
    }
}
```

Anhand der angeforderten Datei aus der HTTP-Anfrage wird entschieden, auf welches CIFS-Laufwerk zugegriffen werden soll. Die erste Komponente im Pfad stellt dabei den Laufwerksnamen dar. Für die Anfrage `/intern/support/antrag.txt` würde also auf die Datei `support/antrag.txt` auf dem Netzlaufwerk `intern` zugegriffen werden. Die Abbildungen der Laufwerksnamen auf die eigentlichen CIFS-Laufwerke übernimmt hierbei die Klasse `ContextMapper`[3]. Sie greift hierfür, wie in Abschnitt 4.4 beschrieben, auf die aktuellen Konfigurationsdaten zu.

4.3 Zugriff auf CIFS-Laufwerke

Zum Zugriff auf Netzlaufwerke nach dem CIFS-Standard wurden die Java-Klassen von [Hranitzky] verwendet. Sie sind im Paket `org.gnu.jcifs` und in Unterpaketen davon enthalten. Der Kern ist das Interface `CifsSession`, dessen Implementierung `CifsDisk` für Verbindungen zu CIFS-Laufwerken eingesetzt wird.

Die Authentifizierung erfolgt durch eine Instanz der Klasse `CifsLogin`, die im Konstruktor von `CifsDisk` übergeben wird. Sie selbst ist lediglich ein Bean, das die beiden Eigenschaften Benutzername und Passwort enthält.

[3] Intern wird in Tomcat die Bezeichnung „Context" für lokale Ressourcen verwendet, daher der Name der Klasse.

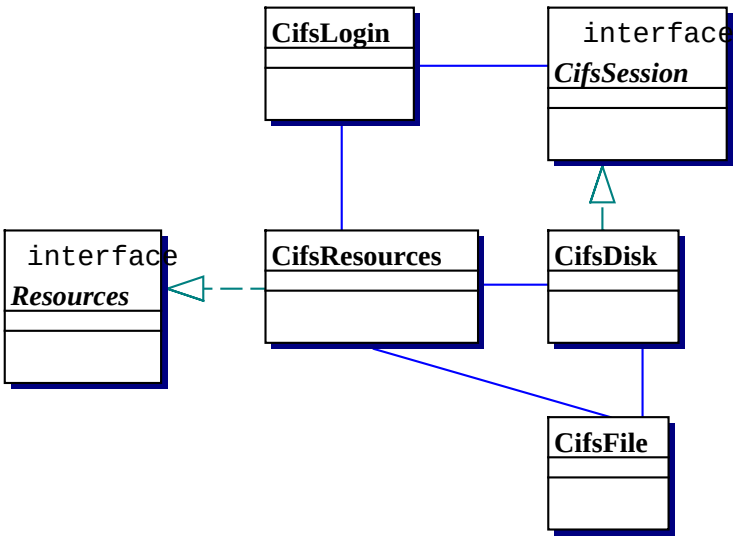
`CifsDisk` bietet eine Methode zur Erzeugung eines `CifsFile`-Objekts, das wiederum eine ganze Reihe von Methoden zum Lesen, Schreiben und Löschen von Dateien, Anlegen von Verzeichnissen und Setzen von Zugriffsberechtigungen zur Verfügung stellt. Als Parameter wird jeweils die Datei beziehungsweise das Verzeichnis als Zeichenkette übergeben.

Das Bindeglied zwischen der `Resources`-Schnittstelle von Tomcat und den Zugriffsmethoden auf `CifsDisk` stellt die Klasse `CifsResources` dar. Sie implementiert das `Resources`-Interface und setzt deren Methoden auf die von `CifsDisk` um. Die `getResourceLength()`-Methode, die zur Ermittlung der Länge der angegebenen Datei dient, hat beispielsweise den folgenden Aufbau. Die Ausnahme (engl. exception) `InvalidPasswordException` wird geworfen, falls das angegebene Passwort in der `CifsLogin`-Instanz ungültig ist. In diesem Fall wird das WebDAV-Servlet nach einem neuen Benutzernamen und zugehörigem Passwort fragen.

```
public long getResourceLength(String path)
  throws InvalidPasswordException, IOException {

  // Zuerst den lokalen Cache durchsuchen
  ResourceBean resource = null;
  synchronized (resourcesCache) {
    resource = (ResourceBean) resourcesCache.get(path);
  }
  if (resource != null) {
    return (resource.getSize());
  }

  // Kein Eintrag im Cache gefunden, nun wird
  // auf die CifsDisk-Instanz zugegriffen
  CifsFile file = new CifsFile(disk, path);
  if (file != null)
    return (file.length());
  else
    return (-1L);

}
```

Den Zusammenhang aller hier besprochenen Klassen und Interfaces zeigt das

folgende Schaubild:



Leider waren in der unter [Hranitzky] erhältlichen Implementierung einige Fehler enthalten, die den Einsatz dieser Klassen mit einem Dateiserver unter Windows NT und 2000 nicht ermöglichten. Mit dem während der Entwicklungsphase verwendeten Samba-Server von [Samba] gab es keinerlei Probleme. Dadurch wurde eine langwierige Fehlersuche, die auch eine direkte Analyse der über das Netzwerk gesendeten Daten beinhaltete, notwendig.

Symptome dieses Fehlers waren bei einem Teil der CIFS-Zugriffe eine Fehlermeldung vom NT-Server. Diese Fehlermeldung mit dem Code `123` war jedoch weder in der offiziellen CIFS-Spezifikation [Leach] noch im Archiv der Samba-Mailingliste [Samba] zu finden. Das Problem lag schließlich in einer unzulässigen Umwandlung von relativen in absolute Pfadnamen. Es wurde noch eine relative Pfadkomponente, dargestellt durch einen Punkt, eingefügt.

Patches zur Behebung dieses und anderer während der Suche gefundener Fehler wurden an den jCIFS-Autor [Hranitzky] geschickt. Sie wurden unter der GNU General Public License, wie in Anhang B beigefügt, veröffentlicht.

4.4 Verwaltung der Konfigurationsdateien

Die Hauptarbeit zur Verwaltung der aktuellen Konfiguration wird von der Klasse `ConfigManager` erledigt. Sie hält eine Liste aller Netzlaufwerke im Speicher und bietet Methoden zum Hinzufügen, Entfernen und Ersetzen von deren Einträgen. Die einzelnen Listenelemente sind Instanzen der Klasse `ConfigElement`, einem Java Bean, das die Eigenschaften eines Netzlaufwerks enthält. Die Namen der Attribute und ihre Bedeutung sind in der nachfolgenden Tabelle aufgelistet:

Attribut	Datentyp	Beschreibung
path	String	Der Pfadname, unter dem das Netzlaufwerk angesprochen werden soll
description	String	Eine Kurzbeschreibung der auf diesem Netzlaufwerk vorhandenen Daten
url	String	Die CIFS-URL zum Zugriff auf das Netzlaufwerk. Sie hat das Format `cifs://host/share`, also zum Beispiel `cifs://filesv1/verwaltung`
domain	String	Der NT-Domainname, in der sich der betreffende Dateiserver befindet. Dieser Wert wird beim Anmelden am Dateiserver dem Benutzernamen vorangestellt.
users	SortedSet	Eine Menge von Benutzernamen[4], die Zugriff auf dieses Netzlaufwerk erhalten sollen. Nur diesen Benutzern wird das Laufwerk in der Übersicht angeboten.

Zum Abspeichern der Konfiguration auf Festplatte wird jede `ConfigElement`-Instanz in einen XML-Ausschnitt umgewandelt. Die einzelnen Attribute der Klasse werden in XML-Tags mit dem gleichen Namen umgewandelt und in-

[4] Die Benutzernamen werden jeweils als String dargestellt.

nerhalb eines `cifsdrive`-Tags gespeichert. Alle `cifsdrive`-Tags werden wiederum nacheinander innerhalb eines `cifsconfig`-Tags, das die Wurzel der XML-Datei darstellt, gespeichert.

Die gerade genannten Aufgaben übernimmt die statische `saveConfig()`-Methode in der Klasse `ConfigWriter`. Der Inhalt der Konfigurationsdatei könnte beispielsweise folgenden Aufbau haben:

```
<?xml version="1.0"?>

<cifsconfig>

  <cifsdrive>
    <path>verwalt</path>
    <description>Netzlaufwerk der Verwaltung</description>
    <url>cifs://fileserv1/verwaltung</url>
    <domain>NT-DOM1</domain>
    <users>tmuster, hbeispiel, theboss</users>
  </cifsdrive>

  <cifsdrive>
    [...]
  </cifsdrive>

  [...]

</cifsconfig>
```

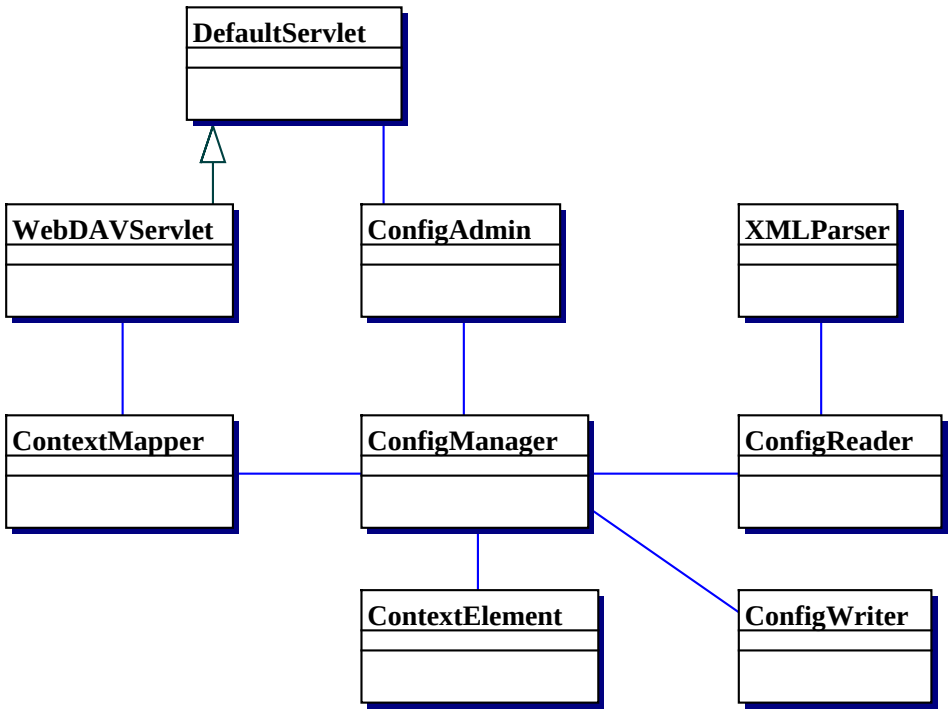
Eingelesen werden kann eine solche Konfigurationsdatei durch die ebenfalls statische `readConfig()`-Methode der Klasse `ConfigReader`. Ihr kann als Parameter ein Dateiname übergeben werden, um verschiedene Konfigurationsdateien zu erlauben. Das Resultat ist beim erfolgreichen Einlesen eine neue Instanz der Klasse `ConfigManager`, die wiederum eine Liste aus `ConfigElement`-Einträgen enthält. Zur Analyse der Datei wird die SAX-Schnittstelle des XML-Parser Xerces[5] eingesetzt. Diese ruft nach dem Callback-Verfahren für jedes gefundene Anfangs- und Ende-Tag die Methoden `startElement()` bzw. `endElement()` auf, innerhalb derer dann die einzelnen `ContextElement`-Beans mit Werten gefüllt werden:

[5] Xerces ist eine Entwicklung des Apache/XML-Projekts und unter einer Open Source-Lizenz im Internet unter [Xerces] verfügbar.

```
// Instanzvariablen

ContextElement ce;
String elemnt;

[...]

public void startElement(String namespaceURI,
                         String localName,
                         String rawName,
                         Attributes atts)
            throws SAXException {

  // In der Instanzvariablen element wird das gerade
  // bearbeitete Tag gespeichert.
  element = localName;

  [...]

  if (localName.equals("cifsdrive") {
    // Beginn eines neuen Netzlaufwerks
    ce = new ContextElement();
  }
}

public void characters(char[] ch, int start, int end)
            throws SAXException {

  // Der Text zwischen Anfangs- und Ende-Tag wird hier
  // im aktuellen ContextElement gespeichert.
  String s = new String(ch, start, end);
  if (element.equals("path") {
    ce.setPath(s);
  }
  if (element.equals("description") {
    ce.setDescription(s);
  }
}
```

Die Klasse `ContextAdmin` bietet schließlich die HTML-Schnittstelle zur grafischen Administration des Systems. Sie holt vom `ContextManager` die Liste aller Netzlaufwerke, stellt diese in einer HTML-Tabelle dar und erlaubt so deren Bearbeitung. Ein Beispiel für diese Darstellung befindet sich im Handbuch für Administratoren in Kapitel 5.2.

Zusammengefasst sind also die im nachfolgenden Schaubild dargestellten Klassen für die Verwaltung der Konfiguration zuständig:



4.5 Authentifizierung über LDAP

Für Zugriffe auf LDAP-Server existieren schon eine Reihe von Klassen, die vom Mozilla-Projekt[6] abstammen. Sie bieten von LDAP-Abfragen bis hin zu

[6]Ziel dieses Projekts ist die Weiterentwicklung des von Netscape veröffentlichten Quellcodes ihres Browsers Communicator. In einem Teilprojekt wurden dabei LDAP-Bibliotheken für diverse Programmiersprachen erstellt und im Internet frei zugänglich gemacht.

Veränderungen der Daten alle Möglichkeiten, die das LDAP-Protokoll selber zur Verfügung stellt.

Mit ihrer Hilfe ist die Authentifizierung von Benutzern durch ein LDAP-Verzeichnis möglich. Dafür musste lediglich eines der in [Mozilla] enthaltenen Beispiele leicht angepasst und eine Methode `checkAuth()` implementiert werden. Diese Methode erwartet als Parameter eine Instanz der Klasse `HttpServletRequest`, aus der alle Daten der HTTP-Anfrage ermittelt werden. Der Benutzername kann direkt durch `getRemoteUser()` gewonnen werden, das Passwort muss aber aus dem `Authorization`:-Header, wie in Abschnitt 3.3.1 beschrieben, ermittelt werden:

```
// Benutzername und Passwort aus dem HTTP-Header
// "Authorization" ermitteln
// Dieses Verfahren funktioniert nur bei der
// basic-Authnetifizierung!

String user = "";
String password = "";
String auth = req.getHeader("Authorization");

if (auth != null) {
  try {

    // Zuerst "Basic " am Anfang abschneiden
    auth = new String(new sun.misc.BASE64Decoder().
                          decodeBuffer(auth.substring(6)));

    // Das Format ist "username:password"
    int index = auth.indexOf(":");
    user = auth.substring(0, index);
    password = auth.substring(index+1);

  } catch (IOException e) {
    // wird ignoriert - wenn sie auftritt, bleiben
    // user und password einfach leer
  }
}
```

Diese beiden Daten werden dann nach dem Verfahren aus 3.6 überprüft. Dazu ist nur ein Methodenaufruf aus den Mozilla-Klassen notwendig. Er liefert `true` zurück, wenn Benutzername und Passwort gültig sind.

Die Authentifizierung über LDAP wird nur verwendet, um die Liste der zugänglichen Netzlaufwerke eines Benutzers zu schützen. Die Daten, die mit dem obigen Beispiel gewonnen wurden, werden außerdem für den Zugriff auf die CIFS-Laufwerke in der Klasse `CifsLogin` verwendet, wie in Abschnitt 4.3 beschrieben.

4.6 Aufteilung in Archive

Um eine einfache Installation, wie in der Aufgabenstellung gefordert, zu ermöglichen, wurden alle benötigten Komponenten in Debian-Pakete[7] aufgeteilt. Eine Java-Laufzeitumgebung ist bereits als Paket unter `ftp://ftp.tux.org/java/debian/` verfügbar, alle anderen Pakete wurden im Laufe dieser Arbeit erstellt.

Einige Bibliotheken sind auch in anderen Projekten gut einsetzbar, daher wurden sie in eigene Pakete verpackt:

Paketname	Version	Beschreibung
`libservlet2.2-java`	2.2-1	Die Servlet-Klassen der Version 2.2 von der Referenz-Implementierung Tomcat [Tomcat]
`libxerces-java`	1.2.0-2	Der XML-Parser Xerces vom Apache/XML-Projekt [Xerces]
`libldap-java`	4.1-1	LDAP-Klassen vom Mozilla Projekt [Mozilla]
`tomcat`	3.2beta6-1	Servlet-Engine mit JSP vom Jakarta-Projekt [Tomcat]

Die Pakete `libservlet2.2-java`, `libxerces-java` und `libldap-java` sind bereits Bestandteil der nächsten Debian-Version und unter

[7] Siehe Abschnitt 3.9

`ftp://ftp.debian.org/dists/woody/` im Internet für alle interessierten Nutzer zugänglich. Das `tomcat`-Paket ist noch nicht auf dem Debian-Server erhältlich, dies wird erst mit Erscheinen der endgültigen Version 3.2 der Fall sein.

Die in dieser Arbeit erstellten Klassen sind in einem Web-Archiv, wie in Abschnitt 3.10.5 erläutert, enthalten. Dieses enthält außerdem die jCIFS-Klassen von [Hranitzky] und kann daher direkt in das Anwendungsverzeichnis von Tomcat kopiert werden. Sofort nach diesem Schritt ist die Anwendung einsatzbereit.

4.6.1 Lizenzen der Pakete

Alle verwendeten Pakete sind unter einer Open Source-Lizenz frei verfügbar. Dies bedeutet, dass all diese Pakete kostenlos sowohl privat als auch kommerziell genutzt werden dürfen. Weiterhin ist der Quellcode verfügbar und darf beliebig – auch verändert – weitergegeben werden. Erst dadurch konnten diese Programme als Teil des Projekts verwendet werden.

`libservlet2.2-java`, `libxerces-java` und `tomcat` sind unter der Apache-Lizenz [ASL 1.1] veröffentlicht, `libldap-java` unter der Netscape-Lizenz [NPL 1.1].

Die erstellten Klassen und die jCIFS-Klassen sind unter der GNU General Public License (GPL) erschienen. Die deutsche Übersetzung dieser Lizenz ist in Anhang B abgedruckt, sie ist jedoch nicht rechtlich verbindlich. Die einzig verbindliche Lizenz ist die englische Originalfassung, die im Internet unter [GPL 2] eingesehen werden kann.

Änderungen an verwendeten Klassen stehen unter der gleichen Lizenz wie die Originalklassen selber. Das heißt, für erweiterte und veränderte Klassen aus dem Tomcat-Projekt gilt die Apache-Lizenz [ASL 1.1], für die Änderungen an der jCIFS-Klassen[8] gilt die GPL [GPL 2].

4.7 Test des Systems

Als erstes wurde der WebDAV-Kern entwickelt, der zuerst lokal mit Hilfe der `FileResources`-Klasse getestet wurde. Dafür wurde das Unix-Programm `cadaver` in der Version 0.13.3 aus dem Debian-Paket von

[8] Siehe Abschnitt 4.3

`ftp://ftp.debian.org/dists/woody/` verwendet, da es zusätzliche Debug-Informationen ausgeben kann.

Nachdem der lokale Zugriff funktionierte, wurde auf einen lokalen Samba-Server über die CIFS-Klassen zugegriffen. Dabei traten sehr wenig Probleme auf, wie folgender Testlauf demonstriert:

```
sgybas@kermit ~> cadaver localhost /dav
Looking up hostname...
Connecting to server... connected.
Authentication required for DAV repository:
Username: sgybas
Password: ******

dav:/dav/> ls
Listing collection '/dav/': collection is empty.

dav:/dav/> put libxerces-java_1.2.0-2_all.deb
Uploading libxerces-java_1.2.0-2_all.deb to
  '/dav/libxerces-java_1.2.0-2_all.deb':
Progress: 100.0% of 637624 bytes succeeded.

dav:/dav/> ls
Listing collection '/dav/': succeeded.
      libxerces-java_1.2.0-2_all.deb   637624  Nov 02 13:11

dav:/dav/> mkdir test1
Creating 'test1': succeeded.

dav:/dav/> ls
Listing collection '/dav/': succeeded.
Coll: test1                                0  Nov 02 13:12
      libxerces-java_1.2.0-2_all.deb   637624  Nov 02 13:11

dav:/dav/> cd test1
dav:/dav/test1/> put libservlet2.2-java_2.2-1_all.deb
Uploading libservlet2.2-java_2.2-1_all.deb to
  '/dav/test1/libservlet2.2-java_2.2-1_all.deb':
Progress: 100.0% of 115992 bytes succeeded.

dav:/dav/test1/> ls
```

```
Listing collection '/dav/test1/': succeeded.
      libservlet2.2-java_2.2-1_all.deb 115992  Nov 02 13:12

dav:/dav/test1/> cd ..
dav:/dav/> rm test1
Deleting 'test1': succeeded.

dav:/dav/> ls
Listing collection '/dav/': succeeded.
      libxerces-java_1.2.0-2_all.deb   637624  Nov 02 13:11

dav:/dav/> exit
Connection closed.
```

Im nächsten Schritt wurde dann ein Zugriff auf einen Dateiserver unter Windows NT versucht, dies scheiterte jedoch, wie schon in Abschnitt 4.3 erwähnt. Nachdem nach mehrwöchiger Fehlersuche auch dies funktionierte, wurde der Windows-Explorer zum WebDAV-Zugriff getestet. Hierbei traten ein paar Warnungen auf Grund von fehlerhaften XML-Antworten auf, diese konnten aber mit Hilfe der Log-Protokolle schnell gefunden werden.

Bis zu diesem Zeitpunkt war die Konfiguration des Systems statisch im Programmcode verankert. Nun ging es also darum, eine XML-Konfigurationsdatei einlesen und schreiben zu können und die Liste aller verfügbaren Netzlaufwerke innerhalb von WebDAV-Zugriffen anzuzeigen.

Diese Liste wurde angezeigt, sobald ein gültiger Benutzername eingegeben wurde, da die LDAP-Authentifizierung noch nicht implementiert war. Zum Zugriff auf Daten von Netzlaufwerken selber war aber ein gültiges Kennwort erforderlich, da diese Überprüfung vom Dateiserver selber vorgenommen wird.

Nachdem alle internen Komponenten geschrieben waren, wurden die HTML-Oberflächen zum Dateizugriff und zur Administration erstellt. Getestet wurden sie mit dem Netscape Navigator 4.75 und dem Internet Explorer 5.0, um die Unabhängigkeit von einem Browser-Hersteller sicherzustellen. Abbildungen solcher Testläufe befinden sich im Benutzerhandbuch in Kapitel 5.1.

4.8 Zusammenfassung

Bei diesem Projekt wurde auf zahlreiche Open Source-Komponenten zurückgegriffen, ohne die es nur mit deutlich höherem Aufwand möglich gewesen wäre. Die folgende Tablle verdeutlicht das Verhältnis zwischen selber entwickelten und wiederverwendeten Klassen:

Projekt	Klassen	Codezeilen
Servlet 2.2	40	11163
Mozilla LDAP 4.1	220	51573
jCIFS 1.1	54	18556
Tomcat 3.2b6	217	56224
Eigene Klassen	37	6634

Die Spalte „Klassen" enthält auch Interfaces und abstrakte Klassen. Bei den eigenen Klassen wurden auch stark veränderte Klassen aus anderen Projekten gezählt, bei der Anzahl der Zeilen aber nur die tatsächlich geänderten Zeilen.

In der Aufstellung wurden alle Klassen dem Tomcat-Projekts berücksichtigt, also auch die komplette Servlet-Engine. Damit entsprechen die Zahlen dem Gesamtsystem wie in Abschnitt 2.2 dargestellt.

Die LDAP-Klassen bieten wesentlich mehr Funktionalität als hier verwendet wurde, deshalb werden für die Summe nur 20 Prozent davon berücksichtigt. Somit besteht das ganze Projekt aus fast 400 Klassen mit ca. 100 000 Codezeilen, davon wurden aber nur ca. 7 Prozent selber entwickelt.

Kapitel 5

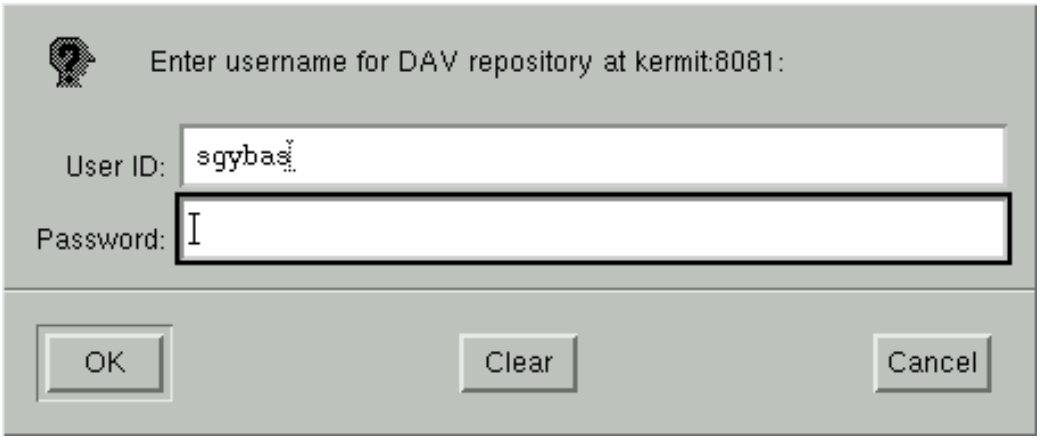
Benutzung des Systems

5.1 Benutzerhandbuch

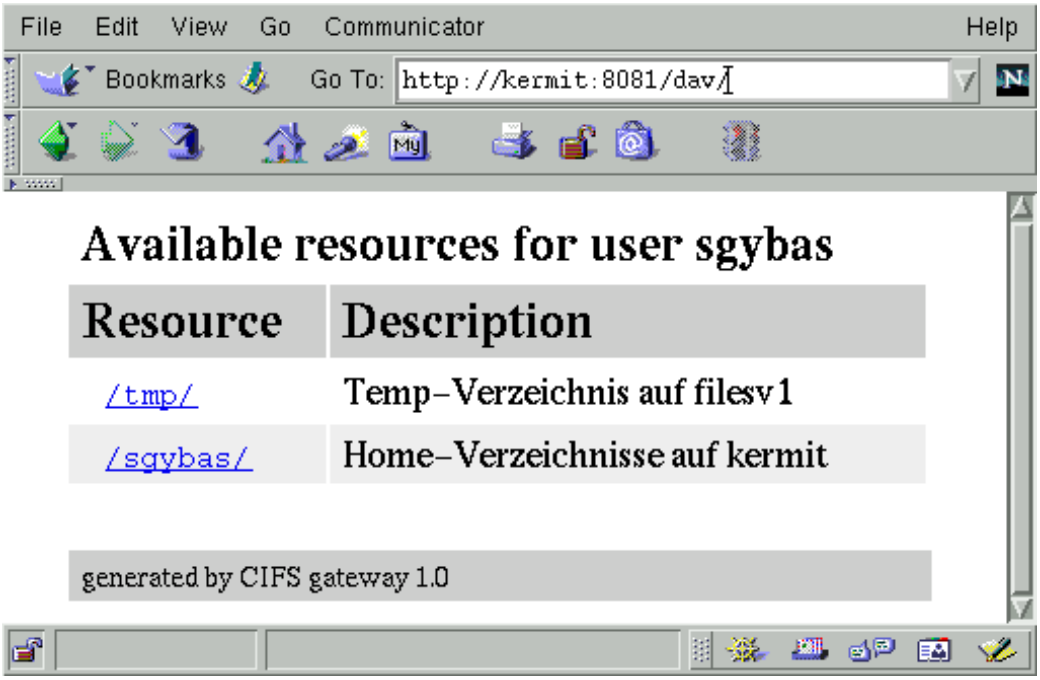
5.1.1 Zugriff mit einem Webbrowser

Die einfachste Zugriffsmethode auf die Netzlaufwerke erfolgt mit einem Webbrowser, zum Beispiel dem Netscape Communicator. Dazu muss in das Adressfeld einfach die URL des WebDAV-Servers eingetragen werden. Hier im Beispiel ist dies `http://kermit:8081/dav/`.

Zuerst werden die Benutzer aufgefordert, ihre Benutzerkennung und ihr Passwort einzugeben:

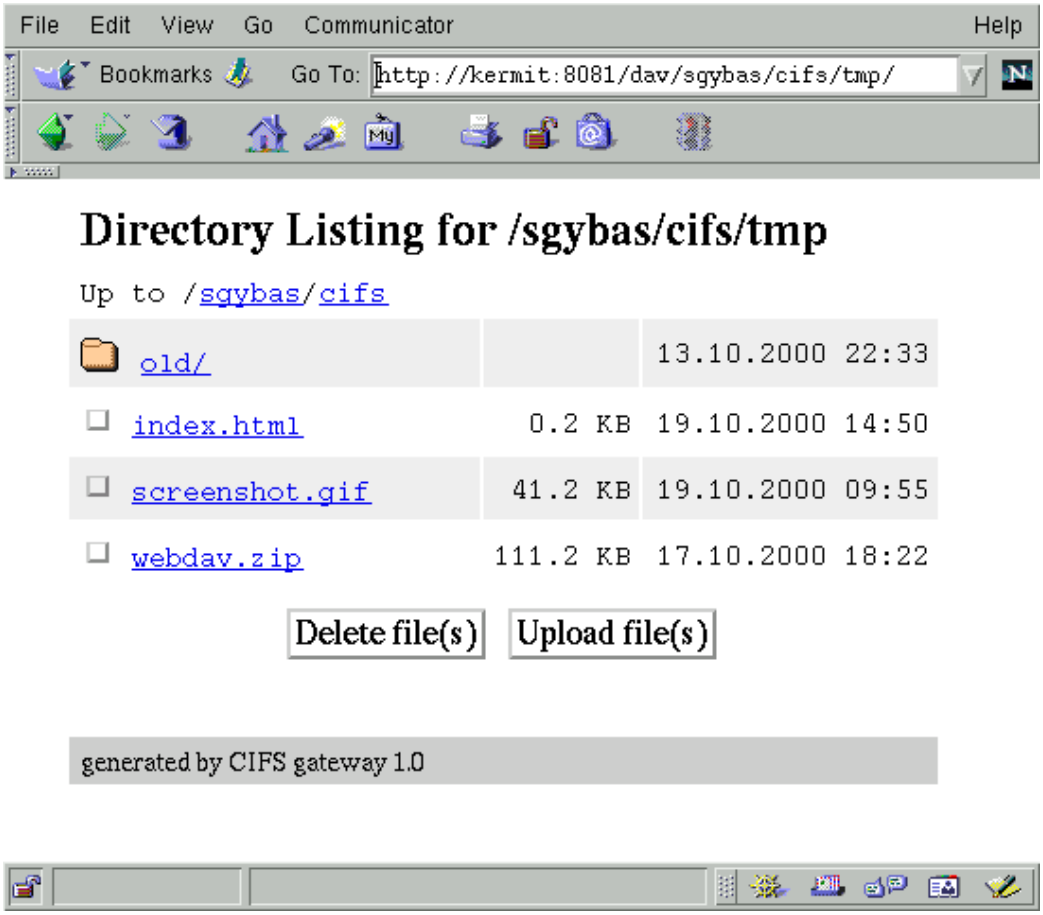


Wenn Benutzername und Passwort gültig sind, wird die Liste der verfügbaren Netzlaufwerke angezeigt:



Durch Anklicken eines Netzlaufwerks erhält man seinen Inhalt als Webseite dargestellt. Diese bietet die Möglichkeit, in ein Verzeichnis zu wechseln, eine Datei herunterzuladen oder eine Datei zu löschen.

Statt die Liste der Netzlaufwerke zu verwenden, kann man auch direkt den Namen des gewünschten Netzlaufwerks an die URL mit anhängen. Für das Netzlaufwerk `sgybas` könnte man also die Adresse `http://kermit:8081/dav/sgybas/` direkt in den Browser eingeben:


File Edit View Go Communicator Help
Bookmarks Go To: http://kermit:8081/dav/sgybas/cifs/tmp/
Directory Listing for /sgybas/cifs/tmp
Up to /sgybas/cifs
old/ 13.10.2000 22:33
index.html 0.2 KB 19.10.2000 14:50
screenshot.gif 41.2 KB 19.10.2000 09:55
webdav.zip 111.2 KB 17.10.2000 18:22
Delete file(s)
Upload file(s)
generated by CIFS gateway 1.0

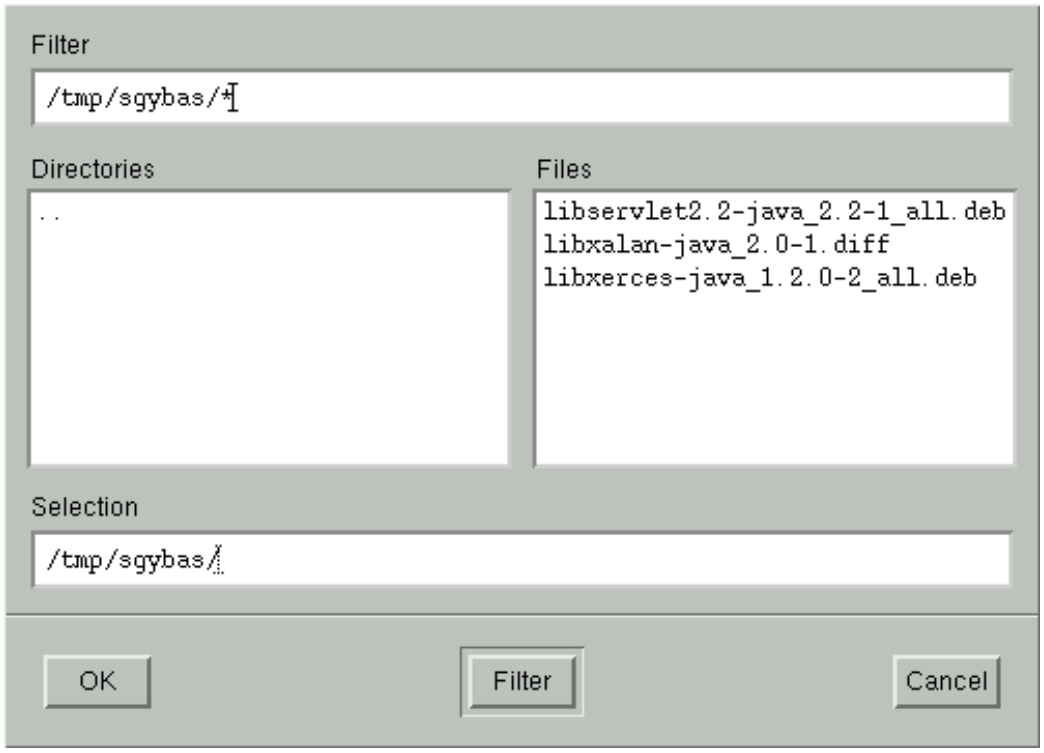
Zum Wechseln in ein Verzeichnis oder zum Herunterladen einer Datei muss einfach nur auf den entsprechenden Eintrag in der Liste geklickt werden. Ganz oben befindet sich noch eine Zeile zum schnellen Zugriff auf alle Oberverzeichnisse, die mit „Up to" gekennzeichnet ist. Hier kann die jeweilige Komponente des Pfadnamens angeklickt werden, um direkt zu diesem Verzeichnis zu springen.

Dateien können gelöscht werden, indem die Checkbox vor ihrem Namen markiert wird und anschließend der Knopf „Delete file(s)" gedrückt wird. Es werden dabei immer alle markierten Dateien gelöscht, eine Sicherheitskopie bleibt jedoch auf dem Backup-Laufwerk[1] erhalten.

Die letzte Möglichkeit dieser Webseite ist das Hochladen von weiteren Dateien. Dazu muss der Knopf „Upload file(s)" betätigt werden, worauf sich ein

[1] Siehe Abschnitt 2.1

Dialog zum Auswählen der Datei öffnet:

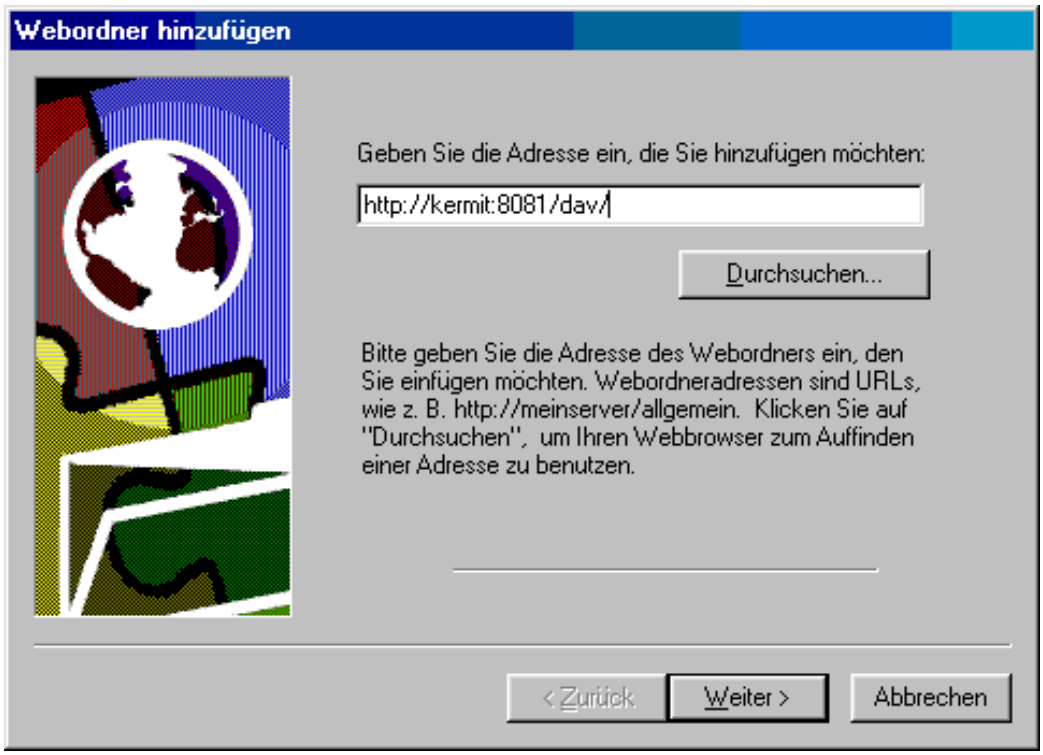


Die gewünschten Dateien können einfach ausgewählt werden. Mit einer Bestätigung auf dem OK-Knopf werden diese dann in das aktuelle Verzeichnis hochgeladen.

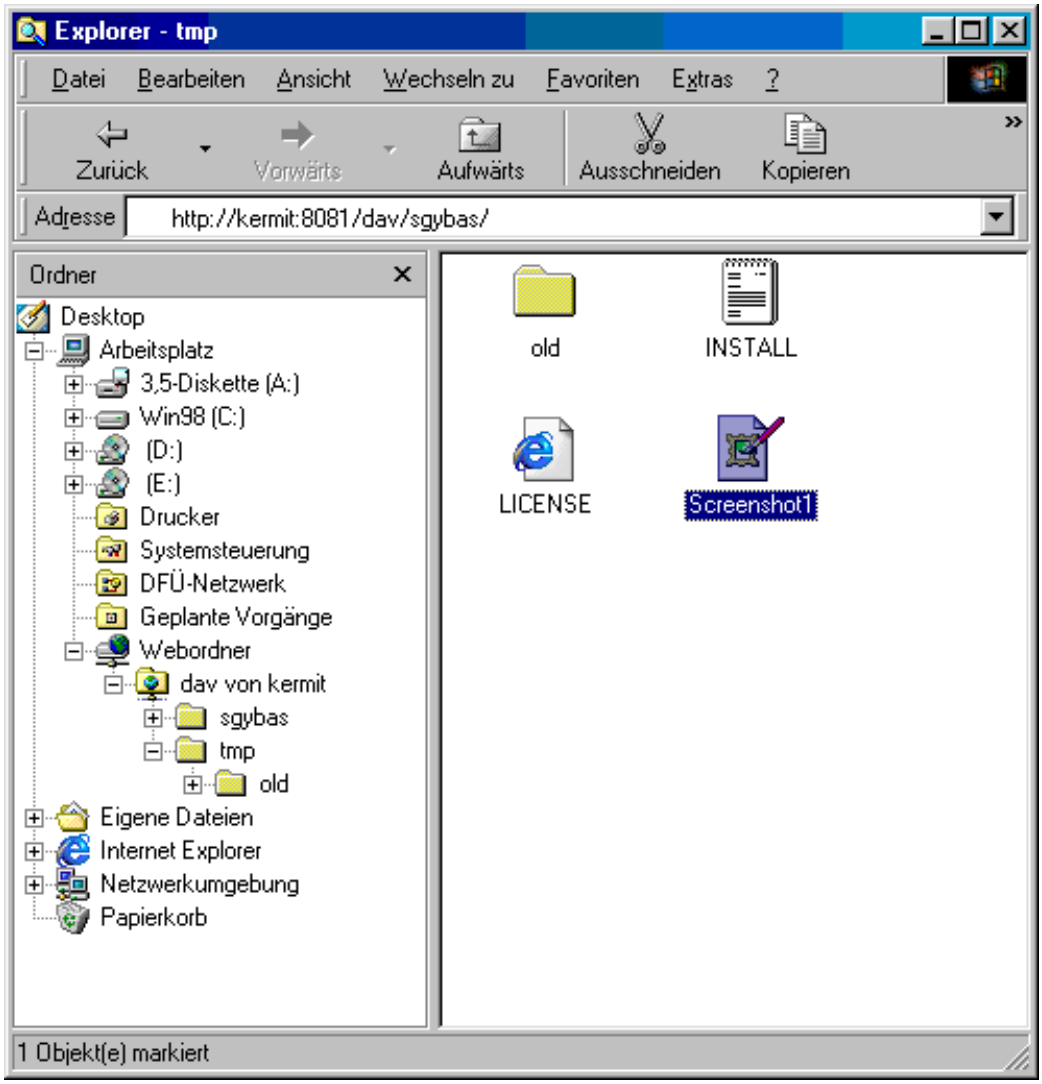
5.1.2 Zugriff mit einem WebDAV-Client

Um weitere Dateioperationen und die Handhabung des Windows-Explorers einsetzen zu können, kann auf die gleiche URL wie beim Webzugriff auch mit WebDAV-Clients zugegriffen werden. Durch die Installation des Internet Explorers 4.0 ist es mit dem normalen Windows-Explorer möglich, direkt auf WebDAV-Laufwerke zuzugreifen. Dort werden diese Laufwerke als „Webordner" bezeichnet.

Zuerst muss ein neuer Webordner verbunden werden:


Webordner hinzufügen
Geben Sie die Adresse ein, die Sie hinzufügen möchten:
http://kermit:8081/dav/
Durchsuchen...
Bitte geben Sie die Adresse des Webordners ein, den Sie einfügen möchten. Webordneradressen sind URLs, wie z. B. http://meinserver/allgemein. Klicken Sie auf "Durchsuchen", um Ihren Webbrowser zum Auffinden einer Adresse zu benutzen.
< Zurück
Weiter >
Abbrechen

Wenn die Verbindung zum WebDAV-Server besteht, kann auf die dortigen Laufwerke wie auf direkt lokal verfügbare Netzlaufwerke zugegriffen werden. Zum Beispiel können Dateien beliebig verschoben oder umbenannt werden oder neue Verzeichnisse angelegt werden:



Um auch Unix-Systemen den WebDAV-Zugang zu ermöglichen, kann das frei erhältliche und schon in Abschnitt 4.7 angesprochene Programm `cadaver` verwendet werden. In diesem Abschnitt befindet sich ebenfalls ein ausführliches Beispiel für dessen Nutzung.

5.2 Administration des Systems

5.2.1 Installation des Systems

Alle benötigten Komponenten sind in einem Web-Archiv enthalten, das nur noch in das Anwendungsverzeichnis einer Servlet-Engine installiert werden muss. Dafür kann jeder Webserver, der die Servlet 2.2- und JSP 1.1-Standards unterstützt, eingesetzt werden.

Die Installation und Konfiguration des Webservers mit Servlet-Engine wird hier am Beispiel der Debian-Pakete `tomcat`, `apache` und `libapache-mod-ssl` vorgeführt. Zuerst sind alle benötigten Pakete mit dem Befehl `apt-get` als root zu installieren. Alle zusätzlich benötigten Pakete werden automatisch mit installiert:

```
# apt-get install apache libapache-mod-ssl \
                  libxerces-java libservlet2.2-java

Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  libssl095a openssl
The following NEW packages will be installed:
  apache libapache-mod-ssl libservlet2.2-java
  libssl095a libxerces-java openssl
Do you want to continue? [Y/n] Y

[...]

(Reading database ... [...]
Unpacking apache (from .../apache_1.3.12-2.1_i386.deb) ...
Selecting previously deselected package libssl095a.
Unpacking libssl095a (from .../libssl095a_0.9.5a-5.deb) ...
Selecting previously deselected package openssl.
Unpacking openssl (from .../openssl_0.9.5a-5.deb) ...
Selecting previously deselected package libapache-mod-ssl.
Unpacking libapache-mod-ssl (from ... [...]) ...
Selecting previously deselected package libservlet2.2-java.
Unpacking libservlet2.2-java (from ... [...]) ...
Selecting previously deselected package libxerces-java.
```

```
Unpacking libxerces-java (from .../libxerces-java_1.2.0-2.deb) ...
Setting up apache (1.3.12-2.1) ...

Setting up libssl095a (0.9.5a-5) ...

Setting up openssl (0.9.5a-5) ...

Setting up libapache-mod-ssl (2.6.4-1.3.12-1) ...
Makefile.crt    ... Skipped
ca-bundle.crt   ... Skipped
server.crt      ... Skipped
snakeoil-ca-dsa.crt ... 0cf14d7d.0
snakeoil-ca-rsa.crt ... e52d41d0.0
snakeoil-dsa.crt ... 5d8360e1.0
snakeoil-rsa.crt ... 82ab5372.0

Setting up libservlet2.2-java (2.2-1) ...

Setting up libxerces-java (1.2.0-2) ...
```

Anschließend müssen die noch nicht im offiziellen Debian-Archiv enthaltenen Pakete `j2sdk` und `tomcat` von Hand installiert werden:

```
# dpkg -i j2sdk1.3_1.3.0-2_i386.deb tomcat_3.1.99b6-1_all.deb

Selecting previously deselected package j2sdk.
Selecting previously deselected package tomcat.
(Reading database ... 124034 files and directories currently installed.)
Unpacking j2sdk (from 2sdk1.3_1.3.0-2_i386.deb) ...
Unpacking tomcat (from tomcat_3.1.99b6-1_all.deb) ...
Setting up j2sdk (1.3.0-2)..

Setting up tomcat (3.1.99b6-1) ...
Starting Tomcat servlet engine: tomcat.
```

Nach diesem Schritt sind alle benötigten Komponenten installiert und vorkonfiguriert. Die Konfiguration des Webservers muss nun noch an die lokalen

Bedürfnisse angepasst werden.

Tomcat enthält zwar selber einen Webserver, doch dessen Leistungsumfang kann mit Apache bei weitem nicht mithalten. Daher wird Apache als Webserver mit SSL-Modul eingesetzt, der alle Anfragen an Java Servlets automatisch an Tomcat weiterleitet. Dazu muss lediglich noch eine von Tomcat erzeugte Datei in die Apache-Konfiguration übernommen werden. Bei dem `tomcat`-Paket liegt diese Datei unter `/var/lib/tomcat/conf/tomcat-apache.conf` vor. Ihr Inhalt muss in den virtuellen Host in der Apache-Konfiguration `/etc/apache/httpd.conf` eingefügt werden. Dies kann zum Beispiel so aussehen:

```
<VirtualHost www.foo.bar:443>
  ServerAdmin   webmaster@foo.bar
  DocumentRoot  /var/www/www.foor.bar/
  ServerName    www.foo.bar
  ErrorLog      /var/log/apache/www.foo.bar-error.log
  TransferLog   /var/log/apache/www.foo.bar-access.log

  SSLEngine     On
  SSLCertificateFile    /etc/apache/ssl.crt/www.foo.bar.crt
  SSLCertificateKeyFile /etc/apache/ssl.key/www.foo.bar.key
  SetEnvIf      User-Agent ".*MSIE.*" nokeepalive [...]

  Include /var/lib/tomcat/conf/tomcat-apache.conf
</VirtualHost>
```

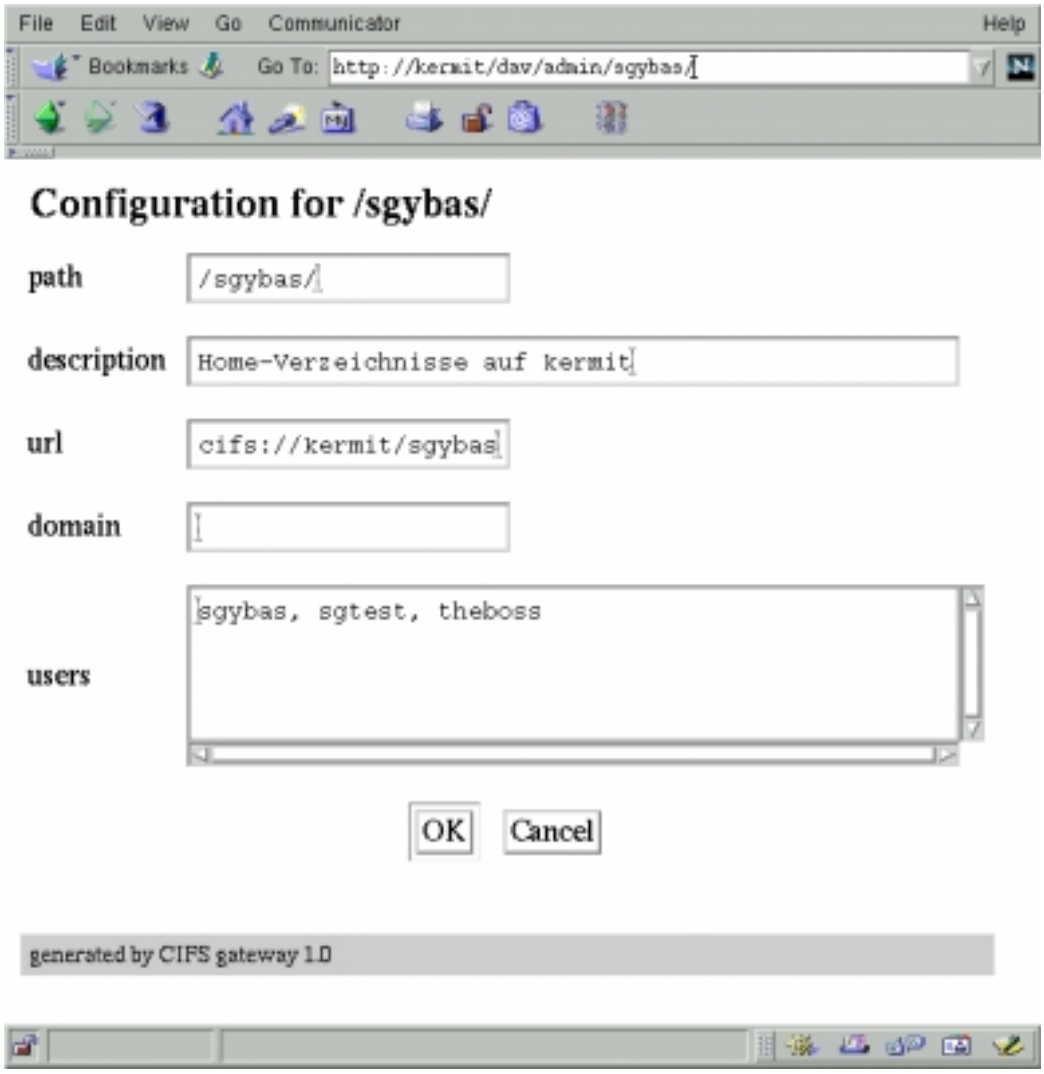
Von nun an ist der virtuelle SSL-Host `www.foo.bar` erreichbar und kann Servlets verwenden.

Zum Schluss muss noch die eigentliche Web-Anwendung installiert werden. Dies geschieht durch einfaches Kopieren des Web-Archivs und Neustarten von Tomcat:

Ab sofort steht die Anwendung unter der Adresse `https://www.foo.bar/dav/` zur Verfügung und kann über eine Web-Oberfläche konfiguriert werden.

5.2.2 Administration über die HTML-Oberfläche

Nach der Grundinstallation aus dem vorherigen Kapitel muss die Anwendung noch konfiguriert werden. Dazu kann mit einem Webbrowser das virtuelle Netzlaufwerk `admin` aufgerufen werden. Im vorherigen Beispiel wäre die Adresse also `https://www.foo.bar/dav/admin/`.



Zuerst erhält man eine Liste aller für Benutzer zugänglichen Netzlaufwerke[2].

[2]Nach der Installation ist diese Liste leer, da ja noch keine Netzlaufwerke konfiguriert wurden.

Hier können bestehende Netzlaufwerke geändert oder gelöscht werden oder durch den Knopf am Ende der Seite ein neues hinzugefügt werden.

Über ein Formular können alle Einstellungen für ein Netzlaufwerk vorgenommen werden. Die Bedeutung der einzelnen Felder kann in der Tabelle in Abschnitt 4.4 nachgelesen werden.

5.2.3 Direkte Bearbeitung der Konfigurationsdatei

Statt über HTML-Formulare wie im vorherigen Kapitel kann auch die XML-Konfigurationsdatei direkt mit jedem beliebigen Texteditor bearbeitet werden. Sie befindet sich im Hauptverzeichnis der Web-Anwendung und trägt den Namen `conf.xml`.

Allerdings muss nach Änderungen die Servlet-Engine neu gestartet werden, damit die Datei neu eingelesen wird. Außerdem ist darauf zu achten, dass die XML-Datei wohlgeformt bleibt, sonst kann der XML-Parser sie nicht mehr lesen. In diesem Fall wird die fehlerhafte Konfigurationsdatei umbenannt und vom WebDAV-Servlet eine neue Konfigurationsdatei ohne Netzlaufwerke erzeugt. Sie kann dann für weitere Konfigurationsarbeiten genutzt werden.

Auf Grund der gerade beschriebenen möglichen Probleme wird davon abgeraten, diese Datei von Hand zu bearbeiten. Es sollte stattdessen immer die Web-Oberfläche genutzt werden, durch die Änderungen außerdem sofort – also ohne einen Neustart der Servlet-Engine – wirksam werden.

Literaturverzeichnis

[ASL 1.1] The Apache Software License, Version 1.1
The Apache Software Foundation, 2000
http://www.apache.org/LICENSE.txt

[CGI 1.1] David Robinson:
The WWW Common Gateway Interface Version 1.1
University of Cambridge, 1996

[CSS 1.0] Håkon Wium Lie, Bert Bos:
Cascading Style Sheets, level 1
World Wide Web Consortium (W3C), 1999
http://www.w3.org/TR/REC-CSS1

[DOM 1.0] Mark Davis et al:
Document Object Model (DOM) Level 2 Core Specification,
Version 1.0
World Wide Web Consortium (W3C), 2000
http://www.w3.org/TR/DOM-Level-2-Core/

[Eckstein] Robert Eckstein, David Collier-Brown, Peter Kelly:
Using Samba, 1st Edition
O'Reilly, 1999

[GPL 2] Richard M. Stallman:
GNU General Public License, Version 2
The Free Software Foundation, 1991
http://www.gnu.org/copyleft/gpl.html

[Hamilton] Graham Hamilton et al:
JavaBeans(TM) API Specification Version 1.01
Sun Microsystems, Inc., 1997
http://java.sun.com/products/javabeans/docs/

[HTML 4.01] Dave Raggett, Arnaud Le Hors, Ian Jacobs:
HTML 4.01 Specification
World Wide Web Consortium (W3C), 1999
http://www.w3.org/TR/html4/

[Hunter] Jason Hunter, William Crawford:
Java Servlet Programming
O'Reilly, 1998

[Hranitzky] Norbert Hranitzky:
Java-Klassen für einen CIFS-Client, Version 1.1
http://www.hranitzky.purespace.de/jcifs/jcifs.htm

[ISO 8879] International Organization for Standardization:
Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML)
ISO-Standard 8879 (Genf), 1986

[ISO 9594] International Organization for Standardization:
Information Processing Systems - Open Systems Interconnection - The Directory: Overview of Concepts, Models and Service (X.500)
ISO-Standard 9594-1 (Genf), 1988

[Jackson] Ian Jackson et al:
Debian Packaging Manual
Software in the Public Interest Inc., 1996
http://www.debian.org/doc/packaging-manuals/

[JSP 1.1] Eduardo Pelegri-Llopart, Larry Cable:
JavaServer Pages(TM) Specification Version 1.1
Sun Microsystems, Inc., 1999
http://java.sun.com/products/jsp/download.html

[Leach] Paul J. Leach, Dilip C. Naik:
A Common Internet File System (CIFS/1.0) Protocol
Microsoft Corporation, 1997
http://www.cifs.com/spec.html

[Mozilla] Miodrag Kekic et al:
Netscape Directory SDK for Java, Version 4.1
The Mozilla Organization, 2000
http://www.mozilla.org/directory/javasdk.html

[Münz] Stefan Münz, Wolfgang Nefzger:
HTML 4.0 Handbuch
Franzis-Verlag, 1998
`http://www.netzwelt.com/selfhtml/`

[NPL 1.1] The Netscape Public License Version 1.1
Netscape Corporation, 1999
`http://www.mozilla.org/MPL/NPL-1.1.html`

[RFC 959] J. Postel, J. Reynolds:
File Transfer Protocol (FTP)
The Internet Engineering Task Force, 1985
`http://www.ietf.org/rfc/rfc959.txt`

[RFC 2045] Ned Freed, Nathaniel S. Borenstein:
Multipurpose Internet Mail Extensions (MIME) Part One:
Format of Internet Message Bodies
The Internet Engineering Task Force„ 1996
`http://www.ietf.org/rfc/rfc2045.txt`

[RFC 2246] T. Dierks, C. Allen:
The TLS Protocol - Version 1.0
The Internet Engineering Task Force„ 1999
`http://www.ietf.org/rfc/rfc2246.txt`

[RFC 2251] M. Wahl, T. Howes, S. Kille:
Lightweight Directory Access Protocol (LDAP v3)
The Internet Engineering Task Force„ 1997
`http://www.ietf.org/rfc/rfc2251.txt`

[RFC 2307] Luke Howard:
An Approach for Using LDAP as a Network Information Service
The Internet Engineering Task Force„ 1998
`http://www.ietf.org/rfc/rfc2307.txt`

[RFC 2396] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter:
Uniform Resource Identifiers (URI): Generic Syntax
The Internet Engineering Task Force„ 1998
`http://www.ietf.org/rfc/rfc2396.txt`

[RFC 2518] Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen:
HTTP Extensions for Distributed Authoring – WEBDAV
The Internet Engineering Task Force„ 1999
`http://www.ietf.org/rfc/rfc2518.txt`

[RFC 2616] R. Fielding et al:
Hypertext Transfer Protocol – HTTP/1.1
The Internet Engineering Task Force„ 1999
http://www.ietf.org/rfc/rfc2616.txt

[RFC 2617] J. Franks et al:
HTTP Authentication: Basic and Digest Access Authentication
The Internet Engineering Task Force„ 1999
http://www.ietf.org/rfc/rfc2617.txt

[RFC 2829] M. Wahl, H. Alvestrand, J. Hodges, R. Morgan:
Authentication Methods for LDAP
The Internet Engineering Task Force„ 2000
http://www.ietf.org/rfc/rfc2829.txt

[Samba] Dr. Andrew Tridgell et al:
The Samba project – an OpenSource SMB/CIFS implementation
http://de.samba.org/samba/samba.html

[SAX 2.0] David Megginson:
SAX (Simple API for XML), Version 2.0
http://www.megginson.com/SAX/

[Schneier] Bruce Schneier:
Applied Cryptography, 2nd Edition
Wiley, 1996

[Servlet 2.2] James Duncan Davidson, Danny Coward:
Java Servlet Specification Version 2.2 Sun Microsystems, Inc., 1999 http://java.sun.com/products/servlet/

[SSL 3.0] Alan O. Freier, Philip Karlton, Paul C. Kocher:
The SSL Protocol Version 3.0
Netscape Corporation, 1996

[Tomcat] The Apache Jakarta Project:
Tomcat: Java Servlet 2.2 and JavaServer Pages 1.1 Reference Implementation, Version 3.2
http://http://jakarta.apache.org/tomcat/index.html

[X.509] X.509 The Directory - Authentication Framework
ITU-T Recommendation, 1988

[Xerces] The Apache XML Project:
Xerces Java XML Parser, Version 1.2.0
`http://xml.apache.org/xerces-j/index.html`

[XHTML 1.0] Dave Raggett et al:
XHTML 1.0: The Extensible HyperText Markup Language
World Wide Web Consortium (W3C), 2000
`http://www.w3.org/TR/xhtml1/`

[XML 1.0] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler:
Extensible Markup Language (XML) 1.0 (Second Edition)
World Wide Web Consortium (W3C), 2000
`http://www.w3.org/TR/REC-xml`

[XML Schema] David C. Fallside:
XML Schema Part 0 to 2
World Wide Web Consortium (W3C), 2000
`http://www.w3.org/XML/Schema`

[XSLT] James Clark:
XSL Transformations (XSLT) Version 1.0
World Wide Web Consortium (W3C), 1999
`http://www.w3.org/TR/xslt`

Anhang A

Ursprüngliche Aufgabenstellung

Ziel dieser Arbeit ist der sichere externe Zugriff von Mitarbeitern auf Daten eines oder mehrerer Projekte. Dabei haben diese Mitarbeiter keinen direkten Netzwerkzugriff auf den Dateiserver mit diesen Daten, z.B. bei einer Einwahl über ISDN werden durch Firewalls nur TCP/IP-Verbindungen zu einem Webserver gestattet.

Es soll daher eine Web-Oberfläche entwickelt werden, die den Benutzern mit Hilfe eines Standard-Webbrowsers den Datenzugriff erlaubt. Diese Oberfläche muß Operationen wie Download, Upload, Löschen, Umbenennen, Verschieben, Setzen von Zugriffsrechten u.s.w. erlauben - also die Datei-Operationen, die auch der NT-Explorer auf dem Windows-Desktop zur Verfügung stellt.

Bei diesem Schema authentifizieren sich die Benutzer zuerst auf einer Webseite durch ihren Benutzernamen und ihr Passwort. Das Benutzerverzeichnis für die Authentifizierung kann dabei von einem LDAP-Server oder einem NT-Domaincontroller stammen. Danach erhalten die Benutzer eine Liste mit allen für sie freigegebenen Verzeichnissen.

Wichtig ist hier, daß die Projektdaten nicht auf dem Webserver selber liegen, sondern auf Dateiservern, auf die der Webserver aber direkten Zugriff (d.h. ohne Firewall) hat. Diese Dateiserver sind i.d.R. NT-Server, die einige Verzeichnisse als Netzlaufwerke freigeben.

Für Administratoren muß eine weitere Web-Oberfläche erstellt werden, die es einfach ermöglichen soll, auf weitere Netzlaufwerke den Zugriff für einzelne Benutzer oder Benutzergruppen zu erlauben oder diesen wieder zu sperren. Außerdem soll es möglich sein, ein Protokoll mit allen Zugriffen sortiert nach Dateiname, Datum oder Benutzer auszuwerten und so Fehlbedienung oder

Mißbrauch zu entdecken.

Um Datenverlust zu vermeiden soll es auch möglich sein, von allen Dateien vor dem Überschreiben bzw. Verändern automatisch eine Sicherheitskopie erstellen zu lassen. Dies kann evtl. über ein Versionsverwaltungssystem realisiert werden.

Die zu erstellende Web-Oberfläche soll unabhängig vom verwendeten Webserver (z.B. Apache oder Netscape) und Betriebssystem (NT oder Unix) einsetzbar sein, daher wird eine Realisierung als Java-Servlets empfohlen. Dadurch ist dann auch ein verschlüsselter SSL-Zugriff von den Webbrowsern auf den Webserver möglich, um die Sicherheit der Projektdaten zu gewährleisten.

Anhang B

GNU General Public License

Diese Übersetzung wird mit der Absicht angeboten, das Verständnis der *GNU General Public License* (GNU-GPL) zu erleichtern. Es handelt sich jedoch nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.

Die *Free Software Foundation* (FSF) ist nicht der Herausgeber dieser Übersetzung, und sie hat diese Übersetzung auch nicht als rechtskräftigen Ersatz für die Original-GNU-GPL anerkannt. Da die Übersetzung nicht sorgfältig von Anwälten überprüft wurde, können die Übersetzer nicht garantieren, daß die Übersetzung die rechtlichen Aussagen der GNU-GPL exakt wiedergibt. Wenn Sie sichergehen wollen, daß von Ihnen geplante Aktivitäten im Sinne der GNU-GPL gestattet sind, halten Sie sich bitte an die englischsprachige Originalversion.

Die *Free Software Foundation* möchte Sie darum bitten, diese Übersetzung nicht als offizielle Lizenzbedingungen für von Ihnen geschriebene Programme zu verwenden. Bitte benutzen Sie hierfür stattdessen die von der *Free Software Foundation* herausgegebene englischsprachige Originalversion.

Es ist jedermann gestattet, diese Lizenzurkunde zu vervielfältigen und unveränderte Kopien zu verbreiten; Änderungen sind jedoch nicht erlaubt.

B.1 Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die *GNU General Public License*, die Allgemeine Öf-

fentliche GNU-Lizenz, ebendiese Freiheit garantieren. Sie soll sicherstellen, daß die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der *Free Software Foundation* herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Datenwerk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden. (Ein anderer Teil der Software der *Free Software Foundation* unterliegt stattdessen der *GNU Library General Public License*, der Allgemeinen Öffentlichen GNU-Lizenz für Bibliotheken.) [Mittlerweile wurde die GNU Library Public License von der GNU Lesser Public License abgelöst – Anmerkung des Übersetzers.]

Die Bezeichnung *„freie" Software* bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, daß Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen – und daß Sie wissen, daß Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbieten, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie – kostenlos oder gegen Bezahlung – Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, daß auch die Empfänger den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüberhinaus sicherstellen, daß jeder erfährt, daß für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, daß die Empfänger wissen, daß sie nicht das Original erhalten haben, damit irgendwelche von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-

Patente bedroht. Wir möchten die Gefahr ausschließen, daß Distributoren eines freien Programms individuell Patente lizensieren – mit dem Ergebnis, daß das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, daß jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muß oder überhaupt nicht lizenziert werden darf.

B.2 Allgemeine Öffentliche GNU-Lizenz

Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung

§ 0. Diese Lizenz gilt für jedes Programm und jedes andere Datenwerk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, daß das Datenwerk unter den Bestimmungen dieser *General Public License* verbreitet werden darf. Im folgenden wird jedes derartige Programm oder Datenwerk als „das Programm“ bezeichnet; die Formulierung „auf dem Programm basierendes Datenwerk“ bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Datenwerk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im folgenden wird die Übersetzung ohne Einschränkung als „Bearbeitung“ eingestuft.) Jeder Lizenznehmer wird im folgenden als „Sie“ angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Datenwerk darstellt (unabhängig davon, daß die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

§ 1. Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, daß Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluß veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und desweiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgeld eine Garantie für das Programm anbieten.

§ 2. Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Datenwerk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, daß zusätzlich alle im folgenden genannten Bedingungen erfüllt werden:

a) Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.

b) Sie müssen dafür sorgen, daß jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.

c) Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, daß es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdruckt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, daß es keine Gewährleistung gibt (oder anderenfalls, daß Sie Garantie leisten), und daß die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muß der Benutzer darauf hingewiesen werden, wie er eine Kopie dieser Lizenz ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muß Ihr auf dem Programm basierendes Datenwerk auch keine solche Meldung ausgeben).

Diese Anforderungen gelten für das bearbeitete Datenwerk als Ganzes. Wenn identifizierbare Teile des Datenwerkes nicht von dem Programm abgeleitet sind und vernünftigerweise als unabhängige und eigenständige Datenwerke für sich selbst zu betrachten sind, dann gelten diese Lizenz und ihre Bedingungen nicht für die betroffenen Teile, wenn Sie diese als eigenständige Datenwerke weitergeben. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen weitergeben, das ein auf dem Programm basierendes Datenwerk darstellt, dann muß die Weitergabe des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf das gesamte Ganze ausgedehnt werden – und somit auf jeden einzelnen Teil,

unabhängig vom jeweiligen Autor.

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Datenwerke in Anspruch zu nehmen oder Ihnen die Rechte für Datenwerke streitig zu machen, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Datenwerken, die auf dem Programm basieren oder unter seiner auszugsweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt auch das einfache Zusammenlegen eines anderen Datenwerkes, das nicht auf dem Programm basiert, mit dem Programm oder einem auf dem Programm basierenden Datenwerk auf ein- und demselben Speicher- oder Vertriebsmedium dieses andere Datenwerk nicht in den Anwendungsbereich dieser Lizenz.

§ 3. Sie dürfen das Programm (oder ein darauf basierendes Datenwerk gemäß Paragraph 2) als Objectcode oder in ausführbarer Form unter den Bedingungen der Paragraphen 1 und 2 kopieren und weitergeben – vorausgesetzt, daß Sie außerdem eine der folgenden Leistungen erbringen:

a) Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muß. Oder:

b) Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen – zu nicht höheren Kosten als denen, die durch den physikalischen Kopiervorgang anfallen –, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:

c) Liefern Sie das Programm zusammen mit dem schriftlichen Angebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objectcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

Unter dem Quelltext eines Datenwerkes wird diejenige Form des Datenwerkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet „der komplette Quelltext“: Der Quelltext aller

im Programm enthaltenen Module einschließlich aller zugehörigen Modulschnittstellen-Definitionsdateien sowie der zur Compilation und Installation verwendeten Skripte. Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft – es sei denn, diese Komponente selbst gehört zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programms oder von Objectcode dadurch erfolgt, daß der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quelltext als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objectcode zu kopieren.

§ 4. Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

§ 5. Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Datenwerke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Datenwerk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Datenwerks.

§ 6. Jedesmal, wenn Sie das Programm (oder ein auf dem Programm basierendes Datenwerk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür

verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

§ 7. Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluß, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Verbreitungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber, zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluß.

Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

§ 8. Wenn die Verbreitung und/oder die Benutzung des Programms in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geographische Begrenzung der Verbreitung angeben, in der diese Staaten ausgeschlossen

werden, so daß die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

§ 9. Die *Free Software Foundation* kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der *General Public License* veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, daß es dieser Lizenz in einer bestimmten Versionsnummer oder „jeder späteren Version“ *(“any later version”)* unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der *Free Software Foundation* veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der *Free Software Foundation* veröffentlicht wurde.

§ 10. Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung anders sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der *Free Software Foundation* steht, schreiben Sie an die *Free Software Foundation*; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Datenwerke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im allgemeinen zu fördern.

B.2.1 Keine Gewährleistung

§ 11. Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – Marktreife oder Verwendbarkeit für einen bestimmten Zweck.

Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

§ 12. In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

Anhang C

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe.

Stuttgart, 14. November 2000

(Stefan Gybas)